UNIVERSITY OF CALIFORNIA
RIVERSIDE

Semantic Analysis and Retrieval of User-Generated Text

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Nhat Xuan Thong Le

September 2019

Dissertation Committee:

Dr. Vagelis Hristidis, Chairperson
Dr. Vassilis Tsotras
Dr. Ahmed Eldawy
Dr. Mariam Salloum

The Dissertation of Nhat Xuan Thong Le is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I am deeply grateful to my advisor, Dr. Vagelis Hristidis, without whose help and support, I would not have been here. His sharp intellect, perseverance and responsiveness have never stopped impressing me throughout my Ph.D. journey. I have learned so much from him the way to find, approach and solve original problems. His endless support and insightful advice are foremost to my academic and professional career.

I would like to thank Dr. Vassilis Tsotras, Dr. Ahmed Eldawy and Dr. Mariam Salloum for being in my committee and for their guidance, constructive feedback and encouragement that help me finish this dissertation. I also thank Dr. Neal Young, Dr. Qi Zhu and Dr. Marko Princevac for joining my committee in my oral qualifying exam.

I thank all my collaborators whose collective knowledge and guidance are essential to this dissertation. I also thank all Database lab members, to whom I have had so many collaborations, fruitful discussions, both academic and non-academic, which spice up my Ph.D. path. Further, we would like to thank all friends I have met and hanged out for all the joy and memorable experience.

Finally, I owe the greatest thanks to my family, whose unconditional love, support and faith have led me to this far.

Acknowledgement of previously published material: Chapter 2 and 3 of this dissertation, in part or in full, is a reprint of the material as it appears in *Proceeding of IEEE 33rd International Conference on Data Engineering, ICDE 2017* ("Ontology-and Sentiment-aware Review Summarization"), *Proceeding of 20th International Conference on Web Information Systems Engineering, WISE 2019* ("Unsupervised Ontology- and Sentiment-aware

Review"), *Proceeding of 13th IEEE International Conference on Semantic Computing, ICSC 2019* ("Targeted Solicitation of Product Reviews"), and the *International Journal of Semantic Computing (IJSC) 13(4), 2019* ("Decrease Product Rating Uncertainty Through Focused Reviews Solicitation").

To my family for all the support.

ABSTRACT OF THE DISSERTATION

Semantic Analysis and Retrieval of User-Generated Text

by

Nhat Xuan Thong Le

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, September 2019
Dr. Vagelis Hristidis, Chairperson

With the ever-increasing volume of user-generated text (e.g., product reviews, doctor notes, chat logs), there is a need to distill valuable semantic information from such unstructured sources. We initially focus on product reviews, which conceptually consist of concepts (or aspects) such as "screen brightness", and user opinions on these concepts such as "very positive". First, we present a novel review summarization framework that advances the state-of-the-art by leveraging a domain hierarchy of concepts to handle the semantic overlap among the aspects, and by accounting for different opinion levels. Second, we argue and empirically show that the current style of soliciting customer opinion by asking them to write free-form text reviews is suboptimal, as few aspects receive most of the ratings. Therefore, we propose various techniques to dynamically select which aspects to ask users to rate given the current review history of a product.

The last body of work leverages user chat logs to continuously optimize the workflow of a goal-oriented chatbot, such as a pizza ordering bot. On one hand, diagram-based chatbots are simple and interpretable but only support limited predefined conversation sce-

narios. On the other hand, the state-of-the-art Reinforcement Learning (RL) models can handle more scenarios but are not interpretable. We propose a hybrid method, which enforces workflow constraints in a chatbot, and uses RL to select the best chatbot response given the specified constraints.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

User-generated text is any free-form text created by users such as online customer reviews, conversational chat logs and forum posts. The volume of this content has dramatically grown thanks to the booming of the Internet 2.0 era. As an example, Amazon.com has millions of products, among which a popular product such as the Echo speaker receives thousands of customer reviews and inquiries. Similarly, the digital assistant systems, for example, Apple Siri, Amazon Alexa and Google Assistant, enable users to interact with machines in their natural language form, thus generate massive data of user chat logs. Further, the rapid adoption of mobile applications and Internet-of-Things devices further increases the amount of user-generated text. There are great challenges and opportunities on organizing, retrieving and mining these big data.

Efficient analysis and retrieval of user-generated text can facilitate many applications beneficial to both the users and the systems generating that content. For instance, online customer reviews can be organized and summarized to provide diverse and thorough

opinions to potential customers, as well as with valuable feedback to the manufactures. In the context of user chat logs, the conversational system can continuously improve its performance using previous user chat sessions. The key barrier to these applications is the unstructured nature of user-generated data.

Since user-generated text is in the form of plain natural language, its analysis bears all challenges that Natural Language Processing (NLP) techniques face including sentiment analysis inaccuracy, various user representations of the same meaning and grammatically improper sentences by online users, to name a few. Furthermore, organizing user-generated data must deal with the redundancy issue, which is two-fold: too much text with overlapping, correlated information and too many users with different personal views of the same subject. Finally, user evaluation with ground truth is not always available. For example, there is no golden standard summary for a product review since summaries are subjective with respect to the individual evaluator; even the same evaluator may have different opinions at different times.

This dissertation extracts the semantic meaning embedded in user-generated text for efficient analysis and retrieval. A first step is to generate a more structured semantic representation of the user-generated text. For instance, in the customer review *"this phone has a very nice screen"*, the user is giving opinion about the "screen" aspect of the phone with a very positive tone. In this dissertation, we study three main questions. We provide short introduction for each part below.

1.1 Online Review Summarization

In this Web 2.0 era, there is an ever increasing number of product and service reviews, which must be summarized to help consumers effortlessly make informed decisions. Previous work on reviews summarization has simplified the problem by assuming that aspects (e.g., "display") are independent of each other and that the opinion for each aspect in a review is Boolean: positive or negative. However, in reality aspects may be interrelated – e.g., "display" and "display color" – and the sentiment takes values in a continuous range – e.g., "somewhat" vs. "very positive".

In Chapter 2, we present a novel review summarization framework that advances the state-of-the-art by leveraging a domain hierarchy of concepts to handle the semantic overlap among the aspects, and by accounting for different sentiment levels. We show that the problem is NP-hard and present bounded approximate algorithms to compute the most representative set of sentences or reviews, based on a principled opinion coverage framework. We experimentally evaluate the proposed algorithms on real datasets in terms of their efficiency and effectiveness compared to the optimal algorithms. We also compare the quality of our summarization methods to multiple baselines, using several intuitive quality measures. The results show that our methods generate summaries of superior quality in short execution times using various intuitive quality measures.

1.2 Targeted Solicitation of Customer Reviews

Customer reviews have become an essential resource when people search for goods or services on the Internet. Previous works [21, 40, 37] have shown that reducing a product's

uncertainty is critical to its purchase decision. Thus, reviews are more effective when they help to reduce a product's uncertainty. Existing e-commerce platforms typically ask users to write free-form text reviews, which are sometimes augmented by a small set of predefined questions, e.g., "rate the product description's accuracy from 1 to 5." In Chapter 3, we argue that this "passive" style of review solicitation is suboptimal in achieving low-uncertainty "review profiles" for products. Its key drawback is that some product aspects receive a very large number of reviews while other aspects do not have enough reviews to draw confident conclusions. Therefore, we hypothesize that we can achieve lower-uncertainty review profiles by carefully selecting which aspects users are asked to rate.

To test this hypothesis, we propose various techniques to dynamically select which aspects to ask users to rate given the current review profile of a product. We use Bayesian inference principles to define reasonable review profile uncertainty measures; specifically, via an aspect's rating variance. We compare our proposed aspect selection techniques to several baselines on several review profile uncertainty measures. Experimental results on two real-world datasets show that our methods lead to better review profile uncertainty compared to aspect selection baselines and traditional passive review solicitations. Moreover, we present and evaluate a hybrid solicitation method that combines the advantages of both active and passive review solicitations.

1.3 Adaptive and Interpretable Dialog Manager for Goal-oriented Chatbot

In the past few years, the advances in deep learning enable high-quality information extraction systems for human conversations. Together with the popularity of mobile devices, this has stimulated the quick adoption of numerous goal-oriented conversational systems, also known as chatbots, such as calendar reminder in Google Assistant, or pizza ordering bot by Dominos. Most of these commercialized chatbots are frame-based dialog systems [9] that model conversation by frames of semantic slots and dialog acts. For instance, a user utterance "I would like a pepperoni pizza" is translated into the semantic level representation: `(dialog_act=inform, slot=pizza_type, value=pepperoni)`.

Previous works on building goal-oriented chatbot's workflow (or dialog manager) fall into two approaches. On one hand, diagram-based chatbots specify a hard-coded sequence of slots to ask users for information, and therefore are simple, interpretable but only support limited predefined conversation scenarios. Moreover, they do not support the notion of optional slots that may be dynamically activated based on the user characteristics. On the other hand, the state-of-the-art Reinforcement Learning (RL) models are able to handle more scenarios and adapt to the user but are not interpretable.

We propose a middle ground, where RL is used to select the best responses, constrained by a space of possible responses, which are represented using a novel data structure, the *chatbot dependency graph*. The RL model learns from past user actions and optimizes the probability of success, e.g., the probability of ordering a pizza. This work is presented in Chapter 4, where we propose a hybrid model, which enforces flow constraints in a chatbot,

and uses RL to select the best chatbot response given the specified constraints.

The key idea of our model is the notion of Dependency Graph (DG), which is created by the bot designer as a basic blueprint specifying a small set of dependencies among slots (both mandatory and optional). Then, our proposed algorithm converts this DG into a RL model that is trained on user chat logs to learn a concrete, personalized workflow. Our model brings together the benefits of both traditional approaches that are interpretability, supporting flexible scenarios and rich semantic. Our evaluation on a real dataset of movie booking domain shows that this is a promising approach. Specifically, our model achieves higher success rate at a faster rate than the RL based approach in the training phase (not applicable to diagram-based chatbots), and outperforms both traditional methods in the testing phase.

Chapter 2

Efficient Ontology- and Sentiment-aware Review Summarization

2.1 Introduction

Online users are increasingly relying on user reviews to make decisions on shopping (e.g., Amazon, Newegg), finding venues (e.g., Yelp, Foursquare), seeking doctors (e.g., Vitals.com, zocdoc.com) and many others. However, as the number of reviews per item grows, especially for popular products, it is infeasible for customers to read all of them, and discern the useful information from them. Therefore, many methods have been proposed to summarize customer opinions from the reviews [31, 22, 50, 12]. They generally either adapt multi-document summarization techniques to choose important text segments [12],

or they extract product concepts (also referred as aspects or attributes in other works), such as "display" of a phone, and customer's opinion (positive or negative) and aggregate them [31, 22, 50].

However, neither of these approaches takes into account the relationship among product's concepts. For example, assuming that we need the opinion summary of a smartphone, showing that the opinions for both *display* and *display color* are very positive is redundant, especially given that we would have to hide other concepts' opinion (e.g., "battery"), given the limited summary size. What makes the problem more challenging is that the opinion of a user for a concept is not Boolean (positive or negative) but can take values from a linear scale, e.g., "very positive", "positive", "somewhat positive", "neutral", and so on. Hence, if "display" has a positive opinion, but "display color" has neutral, the one does not subsume the other, and both should be part of the summary. Further, a more general concept may cover a more specific but not vice versa.

Our key contribution is a novel review summarization framework that accounts for the relationships among the concepts (product aspects), while at the same time supporting various sentiment levels. Specifically, we model our problem as a pairs coverage problem, where each pair consists of a concept and a sentiment value, and coverage is jointly defined on both of them. We show that the problem of selecting the best concepts and opinions to display is NP-hard even when the relationships among the concepts are represented by a Directed-Acyclic-Graph (DAG). For that, we propose bounded approximation algorithms inspired by well-studied graph coverage algorithms.

To summarize, the review summarization framework consists of the following tasks:

(a) Concept Extraction: we build upon existing work for extracting hierarchical concepts (aspects) from reviews. *(b) Sentiment Estimation*: estimate the sentiment of each mentioned concept on a linear scale. *(c) Select k representatives*: depending on the problem variant, a representative is a concept-sentiment pair (e.g., "display"=0.3), or a sentence from a review (e.g., "this phone has pretty sharp display") or a whole review. Our proposed selection algorithms can be used to select representatives at any of these granularities.

Our contributions can be summarized as below:

- We propose a fresh perspective for the review summarization problem that exploits available concept hierarchies and a novel opinion coverage definition. We model the problem as a coverage optimization problem (Section 2.2) and show how to map a set of reviews to our model (Section 2.5.1).
- We prove that the problem is NP-hard and propose several efficient approximation algorithms with guaranteed bounds (Section 2.4).
- We carry out a thorough evaluation on the cost and time of our proposed algorithms. We experimentally evaluate our methods on real collections of online doctor patient reviews, using popular medical concept hierarchies [10], and corresponding concept medical extraction tools [4, 67]. Other extraction tools can be used for other domains, as discussed in the Related Work section (Section 2.6).
- We perform qualitative experiments on both online doctor patient reviews and online cell phone buyer reviews. Using various intuitive summary quality measures, we show that our method outperforms state-of-the-art review summarization methods (Section 2.5.3).

The remainder of this chapter is organized as follows: we present the related work in Section 2.6, and the conclusion in Section 2.7.

2.2 Problem Framework

Define an item (for example, a doctor or a camera) d as a set of reviews, where each review is a set of *concept-sentiment* pairs $\{(c_1, s_1), (c_2, s_2), \ldots, (c_n, s_n)\}$, and $s_j \in \mathbb{R}$ is the sentiment for concept c_j in the review. Sometimes the concepts are explicitly rated (e.g., in `edmunds.com` people assign a star rating for *performance*, *comfort*, *ride comfort*, *rear seats comfort*, and so on), while other times the review is a text paragraph. In the latter case, we use existing review aspect extraction techniques. For the case of doctor reviews, used in our experiments, we use an the MetaMap concept extraction tool [4] to extract medical concepts. In domains without special tools like Metamap, such as e-commerce, we can employ popular unsupervised [62, 61] or supervised [33, 57] aspect extraction methods (for more details see Section 2.6). To estimate the sentiment of each concept, again several existing works can be leveraged. We use a method based on word-embeddings in our experiments.

The set of concepts are related based on a hierarchical *ontology*. Examples of such ontologies include WordNet [53], BabelNet [58] and ConceptNet [71]. For instance, the "part-whole" relation in those ontologies can be utilized to create the hierarchy of aspects suitable for our framework. Alternatively, Kim et al. [39] automatically extract an aspect-sentiment hierarchy using a Bayesian non-parametric model. In our experiments, we use the SNOMED CT [32] ontology, which is popular in the health domain (Figure 2.1).

We define the (directed) *distance* $d(p_1, p_2)$ between two concept-sentiment pairs

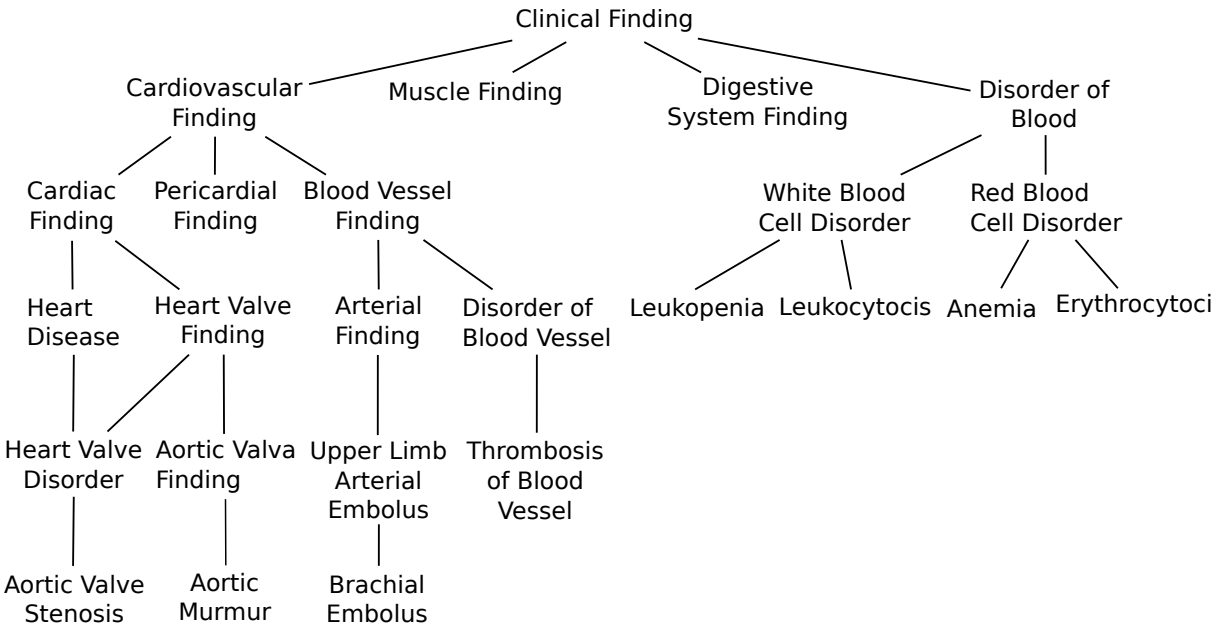


Figure 2.1: SNOMED CT's concept hierarchy sample

$p_1 = (c_1, s_1)$ and $p_2 = (c_2, s_2)$, based on the concepts' relationship in the hierarchy, as follows.

Definition 1 *The distance* $d(p_1, p_2)$ *is:*

$$d(p_1, p_2) = \begin{cases} d(r, c_2) & \textit{if } c_1 \textit{ is the root } r, \textit{ or} \\ d(c_1, c_2) & \textit{if } c_1 \textit{ is the ancestor of } c_2 \\ & \textit{and } |s_1 - s_2| \leq \epsilon, \textit{ or} \\ \infty & \textit{otherwise} \end{cases}$$

where the distance between two concepts $d(c_1, c_2)$ *is the shortest-path length from* c_1 *to* c_2 *in the hierarchy,* r *is the root of the hierarchy, and* $\epsilon > 0$ *is the sentiment threshold.*

If pair p_1 has finite distance to p_2, we say that p_1 *covers* p_2. Pair p_1 covers p_2 iff p_1's concept c_1 is an ancestor of p_2's concept c_2, and either c_1 is the root concept or the sentiments of

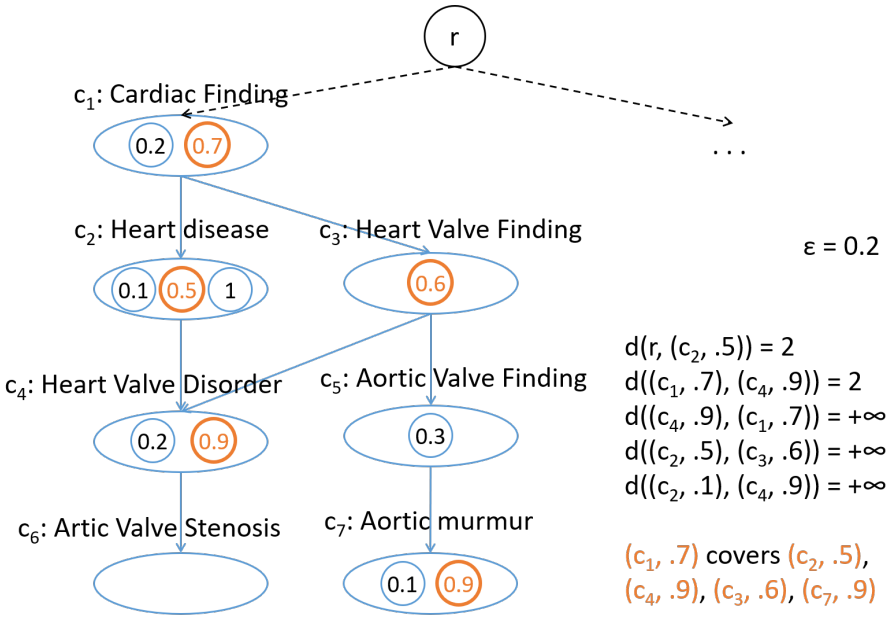


Figure 2.2: Representation of concept-sentiment pairs on the concept hierarchy DAG of Figure 2.1.

p_1 and p_2 differ by at most ϵ. Figure 2.2 shows an example of how the concept-sentiment pairs of an item's reviews are mapped on the concept hierarchy, where the dashed line is the path from the root, and concept c_6 doesn't have any pairs. For instance, pair $(c_1, 0.7)$ represents an occurrence of concept c_1 in a review with sentiment 0.7. The same pair is also represented by the circled 0.7 value inside the c_1 tree node.

Given a set $P = \{p_1, p_2, \ldots, p_q\}$ of concept-sentiment pairs for the reviews of an item, and an integer k, our goal is to compute a set $F = \{f_1, f_2, \ldots, f_k\} \subseteq P$ of k pairs that best summarize P. To measure the quality of such a summary F, we define its cost $C(F, P)$ as the distance from F to P, defined as follows.

Definition 2 *The distance from F to a pair p is the distance of the closest pair in $F \bigcup \{r\}$ to p:* $d(F, p) = \min_{f \in F \bigcup \{r\}} d(f, p)$.

The cost of F is the sum of its distances to pairs in P: $C(F,P) = \sum_{p \in P} d(F,p)$.

We introduce two summarization problems on a set of concept-sentiment pairs:

1. ***k-Pairs Coverage:*** given a set P of concept-sentiment pairs (coming from a given set of reviews for an item) and integer $k \leq |P|$, find a subset $F \subseteq P$ with $|F| = k$ that summarizes P with minimum cost:

$$\min_{F \subseteq P, |F|=k} C(F,P)$$

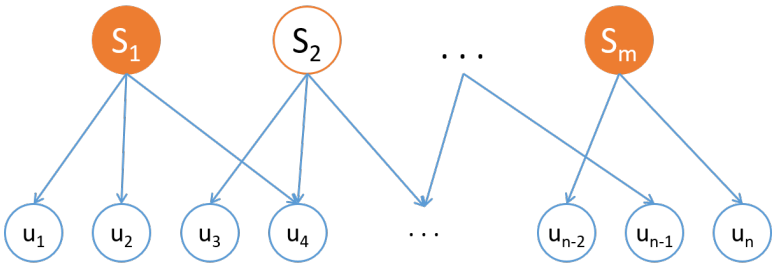
2. ***k-Reviews/Sentences Coverage:*** given a set R of reviews (or sentences) and integer $k \leq |R|$, find a subset $X \subseteq R$ with $|X| = k$ that summarizes R with minimum cost:

$$\min_{X \subseteq R, |X|=k} C(P(X), P(R)),$$

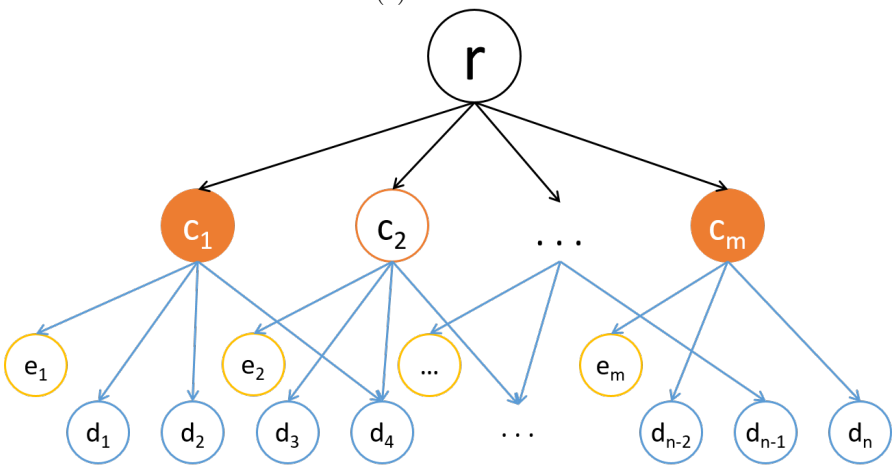
where $P(R)$ is the set of concept-sentiment pairs derived from the set R of reviews/sentences, and $P(X)$ is the set of concept-sentiment pairs derived from the subset X of R.

Intuitively, the first problem is appropriate when the summaries consist of concise concept-sentiment pairs, e.g. "good Heart Disease management", extracted from the reviews, and may be more suitable for mobile phone-sized screens. The second problem is appropriate if the summaries consist of whole sentences of reviews, which better preserves the meaning of the review, but may require more space to display.

The k-Pairs Coverage problem can be viewed as a special case of the k-Reviews/Sentences Coverage problem, when each review/sentence has just one pair. For presentation simplicity, we first present our NP-hard proof and algorithms for k-Pairs Coverage in Section 2.4, then describe how they can be applied to the k-Reviews/Sentences Coverage in Section 2.4.5.



(a) Set Cover



(b) Corresponding instance of k-Pairs Coverage

Figure 2.3: Reduction from Set Cover

2.3 Both Problems are NP-Hard

This section proves both proposed problems NP-hard.

Theorem 3 *The k-Pairs Coverage problem is NP-hard.*

Proof.

The decision problem is, given a set P of concept-sentiment pairs, an integer $k \leq |P|$, and a target $t \geq 0$, to determine whether there exists a subset $F \subseteq P$ of size k with cost $C(F, P)$ at most t. We reduce Set Cover to it. Fix any Set-Cover instance (S, U, k)

where U is the universe $\{u_1, u_2, \ldots, u_n\}$, and $S = \{S_1, S_2, \ldots, S_m\}$ is a collection of subsets of U, and $k \leq |S|$. Given (S, U, k), first construct a concept-hierarchy (DAG) with root r, concepts c_i and e_i for each subset S_i, and a concept d_j for each element u_j. For each set S_i, make c_i a child of r and e_i a child of c_i. For each element u_j, make d_j a child of c_i for each set S_i containing u_j. (See Figure 2.3.) Next, construct $2m+n$ concept-sentiment pairs $P = \{p_1, \ldots, p_{2m+n}\}$, one containing each node in the DAG other than the root r, and all with the same sentiment, say 0. Take target $t = 3m+n-2k$. This completes the reduction. It is clearly polynomial time. Next we verify that it is correct. For brevity, identify each pair with its node.

Suppose S has a set cover of size k. For the summary $F \subseteq P$ of size k, take the k concepts in P that correspond to the sets in the cover. Then each d_i has distance 1 to F, contributing n to the cost. For each set in the cover, the corresponding c_i and e_i have distance 0 and 1 to F, contributing k to the cost. For each set not in the cover, the corresponding c_i and e_i have distance 1 and 2 to F, contributing $3(m-k)$ to the cost, for a total cost of $n + 3m - 2k = t$.

Conversely, suppose P has a summary of size k and cost $t = n + 3m - 2k$. Among size-k summaries of cost at most t, let F be one with a maximum number of c_i nodes. We show that the sets corresponding to the (at most k) c_i nodes in F form a set cover. Assume some $c_{i'}$ is missing from F (otherwise $k \geq m$ so we are done). For every e_i in F, its parent c_i is also in F. (Otherwise adding c_i to F and removing e_i would give a better summary F', i.e., a size-k summary of cost at most t, but with more c_i nodes than F, contradicting the choice of F). No e_i is in F (otherwise removing e_i and adding the missing node $c_{i'}$ would

give a better summary F'). No d_j is in F (otherwise, since neither $e_{i'}$ nor $c_{i'}$ are in F, removing d_j from F and adding $c_{i'}$ would give a better summary F'). Since no e_i or d_j is in F, only c_i nodes are in F. Since the cost is at most $t = n + 3m - 2k$, by calculation as in the preceding paragraph, the sets S_i corresponding to the nodes c_i in F must form a set cover. ■

When we already have k-Pairs Coverage as a NP-hard problem, it's natural to prove the following theorem.

Theorem 4 *The k-Reviews/Sentences Coverage problem is NP-hard.*

Proof. K-Reviews/Sentences Coverage is a generalization of k-Pairs Coverage, so the theorem follows from the previous theorem. ■

2.4 Algorithms

We implement three algorithms for k-Pairs Coverage. The first, which is the only one generates an optimal solution, solves the standard integer-linear program (ILP) for the problem, as a special case of the well-known k-Medians problem. The second randomly solves the linear program (LP), then randomly rounds the fractional solution achieving a bounded approximation error. The third is a greedy bounded approximation algorithm. The three algorithms share a common initialization phase that we describe first.

2.4.1 Initialization

The initialization phase computes the underlying edge-weighted bipartite graph $G = (U, W, E)$ where vertex sets U and W are the concept-sentiment pairs in the given set

P, edge set E is $\{(p, p') \in U \times W : d(p, p') < \infty\}$, and edge (p, p') has weight equal to the pair distance $d(p, p')$. The initialization phase builds G in two passes over P. The first pass puts the pairs $p = (c, s)$ into buckets by category c. The second pass, for each pair $p = (c, s)$, iterates over the ancestors of c in the DAG (using depth-first-search from c). For each ancestor c', it checks the pairs $p' = (c', s')$ in the bucket for c'. For those with finite distance $d(p, p')$, it adds the corresponding edge to G.

For our problems, the time for the initialization phase and the size of the resulting graph G are roughly linear in $|P|$, because the average number of ancestors for each node in the DAG is small.

2.4.2 ILP for optimal solution

Given the graph $G = (U, W, E)$, Figure 2.4 shows the standard k-Medians ILP adapted for our non-standard cost function. Our first algorithm solves the ILP using the Gurobi solver. Of course, no worst-case polynomial-time bounds are known for solving this NP-hard ILP, but on our instances the algorithm finishes in reasonable time (Details are in Section 2.5).

2.4.3 Randomized rounding

The second algorithm computes an optimal fractional solution (x, y) to the LP relaxation of the ILP (using Gurobi, details in Section 2.5), then randomly rounds it as shown in algorithm 1: it chooses the summary F by sampling k pairs p at random from the distribution $x/\|x\|_1$.

$$
\begin{aligned}
\text{minimize} \quad & \sum_{(p,q)\in E} y_{pq} \times d(p,q) \\
\text{subject to} \quad & x_r = 1; \qquad \textstyle\sum_{p\in P\setminus\{r\}} x_p = k; \\
& \textstyle\sum_{\forall q\in W, p:(p,q)\in E} y_{pq} = 1; \\
& (\forall (p,q) \in E \quad 0 \leq y_{pq} \leq x_p; \\
& (\forall p \in U) \quad x_p \in \{0,1\}
\end{aligned}
$$

Figure 2.4: k-Medians ILP

Algorithm 1 Randomized Rounding Algorithm

Input: fractional solution x, y

Output: summary F

1: **procedure** RANDOMIZED ROUNDING

2: Define probability distribution q on $P' = P \setminus \{r\}$

3: such that $q(p) = x_p / \sum_{p\in P'} x_p$.

4: $F = \emptyset$

5: **while** $|F| < k$ **do**

6: Sample one pair p without replacement from q.

7: Add p to F.

8: Return F.

No good worst-case bounds are known on the time to solve the LP, but on our instances the solver solves it in reasonable time. The randomized-rounding phase can easily be implemented to run in linear time, $O(n)$ where $n = |P|$.

This randomized-rounding algorithm is due to [86] (see also [15]). The following worst-case approximation guarantee holds for this algorithm, as a direct corollary of the analysis in [15]. Let $\text{OPT}_k(P)$ denote the minimum cost of any size-k summary of P.

Theorem 5 *The expected cost of the size-k summary returned by the randomized-rounding algorithm is $O(\text{OPT}_{k'}(P))$ for some $k' = O(k/\log n)$.*

In our experiments it gives near-optimal summary costs.

2.4.4 Greedy algorithm

The greedy algorithm is Algorithm 2. It starts with a set $F = \{r\}$ containing just the root. It then iterates k times, in each iteration adding a pair $p \in P$ to F chosen to minimize the resulting cost $C(F \cup \{p\}, P)$. Finally, it returns summary $F \setminus \{r\}$. This is essentially a standard greedy algorithm for k-medians. Since the cost is a submodular function of P, the algorithm is a special case of Wolsey's generalization of the greedy set-cover algorithm [85].

After the initialization phase, which computes the graph $G = (U, W, E)$, the algorithm further initializes a max-heap for selecting p in each iteration. The max-heap stores each pair p, keyed by $\delta(p, F) = C(F \cup \{p\}, P) - C(F, P)$. The max-heap is initialized naively, in time $O(m + n \log n)$ (where $m = |E|$, $n = |P|$). (This could be reduced to $O(m + n)$ with the linear-time build-heap operation.) Each iteration deletes the pair p with maximum key

from the heap (in $O(\log n)$ time), adds p to F, and then updates the changed keys. The pairs q whose keys change are those that are neighbors of neighbors of p in G. The number of these updates is typically $O(d^2)$, where d is the typical degree of a node in G. The cost of each update is $O(\log n)$ time. After initialization, the algorithm typically takes $O(kd^2 \log n)$ time. In our experiments, our graphs are sparse (a typical node p has only hundreds of such pairs q), and k is a small constant, so the time after initialization is dominated by the time for initialization. The following worst-case approximation guarantee is a direct corollary of

Algorithm 2 Greedy Algorithm

Input: $G = (U, W, E)$ from initialization, computed from P.

Output: Size-k summary F.

```
1: procedure GREEDY
2:     Define δ(p, F) = C(F ∪ {p}, P) − C(F, P).
3:     Let F = {r}.
4:     Initialize max-heap holding p ∈ U keyed by δ(p, F).
5:     while |F| < k + 1 do
6:         Delete p with highest key from max-heap.
7:         Add p to F.
8:         for w such that (p, w) ∈ E do
9:             for q such that (q, w) ∈ E do
10:                Update max-heap key δ(q, F) for q.
11:    return F \ {r}
```

Wolsey's analysis [85]. Let $H(i) = 1+1/2+\cdots+1/i \approx 1+\log i$ be the ith harmonic number. Let Δ be the maximum depth of the concept DAG.

Theorem 6 *The greedy algorithm produces a size-k summary of cost at most* $\text{OPT}_{k'}(P)$*, where* $k' = \lfloor k/H(\Delta n) \rfloor$.

In our experiments, the algorithm returns near-optimal size-k summaries.

2.4.5 Adaptation for k-Reviews/Sentences Coverage problem

When whole reviews or sentences (each containing a set of concept-sentiment pairs) must be selected, the above algorithms can still be applied with only a modification of the initialization stage . In particular, we modify the construction of bipartite graph $G = (U, W, E)$, so instead of having both U and W be concept-sentiment pairs in P, U represents the set of candidate reviews or sentences R, and W represents concept-sentiment pairs as before. Therefore the edge set E becomes $\{(r, p) \in U \times W : d(r, p) < \infty\}$, and edge (r, p) has weight equal to the distance $d(r, p)$ from review/sentence r to pair p. After this initialization, the algorithms work as usual.

2.5 Experimental Evaluation

In this section we conduct both quantitative and qualitative evaluations. The quantitative evaluation measures the time and accuracy trade-offs of the proposed approximate summarization algorithms compared to the optimal solution. The qualitative evaluation evaluates the quality of the summaries generated by the proposed methods, compared to baseline state-of-the-art summarization methods using several intuitive measures.

2.5.1 Experiment Setup

Datasets

We utilize two real-world datasets of two different domains: health care and online consumer reviews. Our first dataset consists of 68,686 patient reviews of the 1000 most reviewed doctors from *vitals.com*, which is a popular doctor rating website. As the second dataset, we crawled customer reviews of 60 unlocked cell phones, which are featured in the first five pages on Amazon and have at least 100 distinct reviews each. Table 2.1 presents basic statistics of the two datasets.

Table 2.1: Dataset characteristics.

	Doctor Reviews	Cell Phone Reviews
#Items (doctor/product)	1000	60
#Reviews	68686	33578
Min #reviews per item	43	102
Max #reviews per item	354	3200
Average #sentences per review	4.87	3.81

Concepts and sentences extraction

To extract medical concepts in doctor reviews dataset we use MetaMap [4], which is an automated tool for mapping biomedical text to medical concepts from Unified Medical Language System (UMLS) Metathesaurus [10]. UMLS contains multiple medical ontologies; we choose SNOMED CT [32], which has more than 300,000 concepts and is suitable for our problem given its focus on describing medical conditions. For example, for sentence *"Dr Robert did an awesome job with my tummy tuck and liposuction"*, concepts *"tummy tuck"*

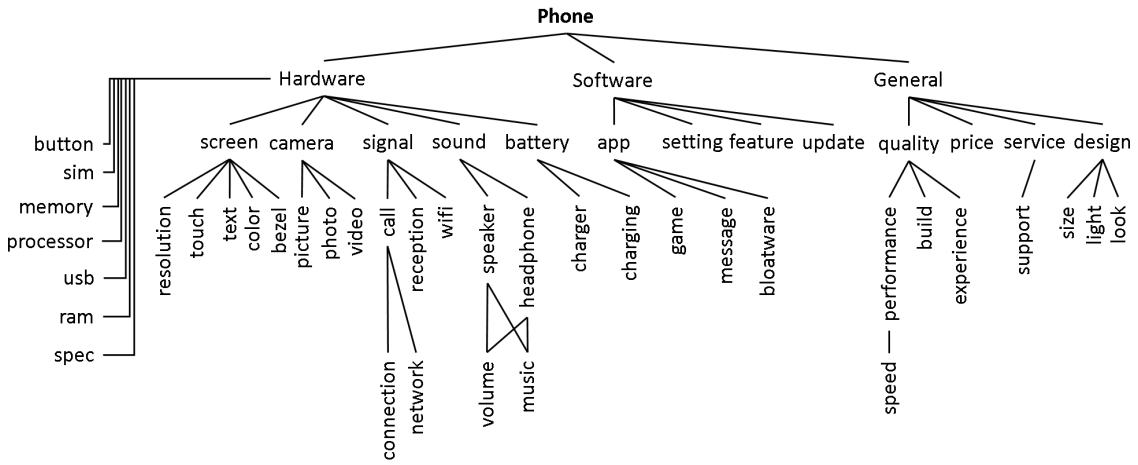


Figure 2.5: Cell phone aspect hierarchy

(UMLS ID=C0749734) and *"liposuction"* (UMLS ID=C0038640) are extracted.

In cell phone reviews dataset, we employ Double Propagation method [62] to extract cell phone aspects such as screen and battery. We only focus on the 100 most popular extracted aspects. Given that there is no available hierarchy of cell phone aspects, we manually built a hierarchy from the extracted aspects as shown in Figure 2.5.

Sentiment Computation

To compute the sentiment around a concept, we first need to define the context, i.e. the *containing sentence*. For that we use the PTBTokenizer library in Stanford POS tagger [78], which generally splits text based on punctuation marks. Then, we compute the sentiment of the containing sentence and assign this sentiment to the concept.

To compute the sentiment of a sentence we adopt a neural network based representation learning approach *doc2vec*, which represents words, sentences, and generally text by fixed-size vectors [44]. Then, sentence's sentiment estimation is formulated as a standard

regression problem using the sentence vector representation.

Configuration

We evaluate the three methods proposed in Section 2.4: Integer Linear Programming - ILP, Randomized Rounding - RR, and Greedy algorithm. For ILP and RR, we use the Gurobi optimization library version 6.0.5 [29] with Dual-Simplex as the default method. This method is chosen because it shows the best performance in our case after experimental trials on different options available in Gurobi (primal simplex, barrier, auto-switching between methods, concurrent). All experiments were executed on a single machine with Intel Core i7-4790 3.60 GHz, 16 GB RAM, Windows 10 professional 64 bits. Our code was written in Java using the official Oracle Java version 8 update 45.

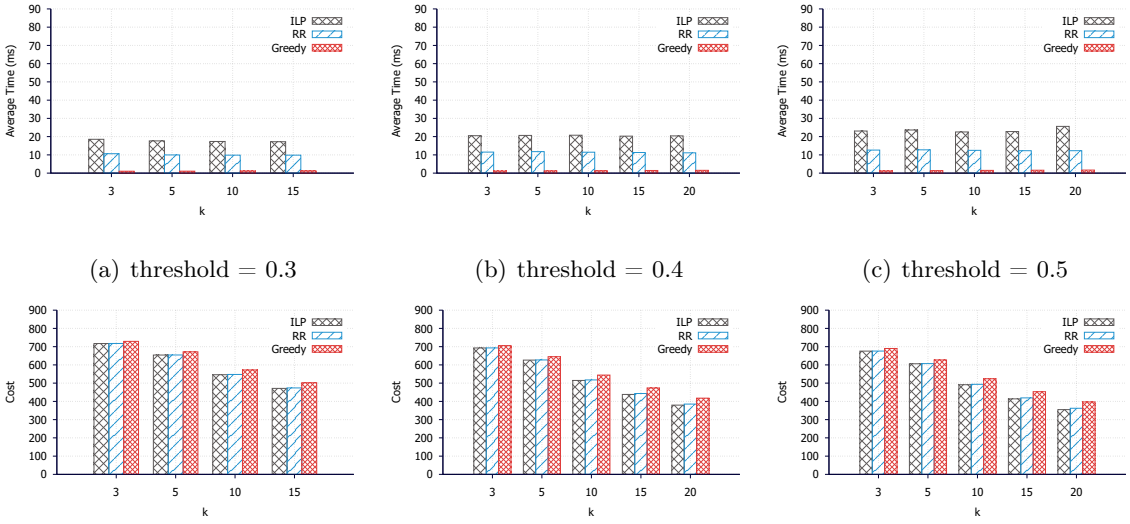
2.5.2 Quantitative Evaluation

For brevity, we only present results on doctor reviews dataset, which is the larger dataset, in this section.

Average Coverage Cost and Time of Algorithms

We compare the average coverage cost (defined in Definition 2) and time of our three algorithms. They are evaluated on the problems of finding top pairs, sentences and reviews, in Figures 2.6, 2.7 and 2.8 respectively. These figures show the elapsed time (in ms) and cost of the algorithms when the threshold on sentiment coverage definition is varied from 0.3 to 0.5.

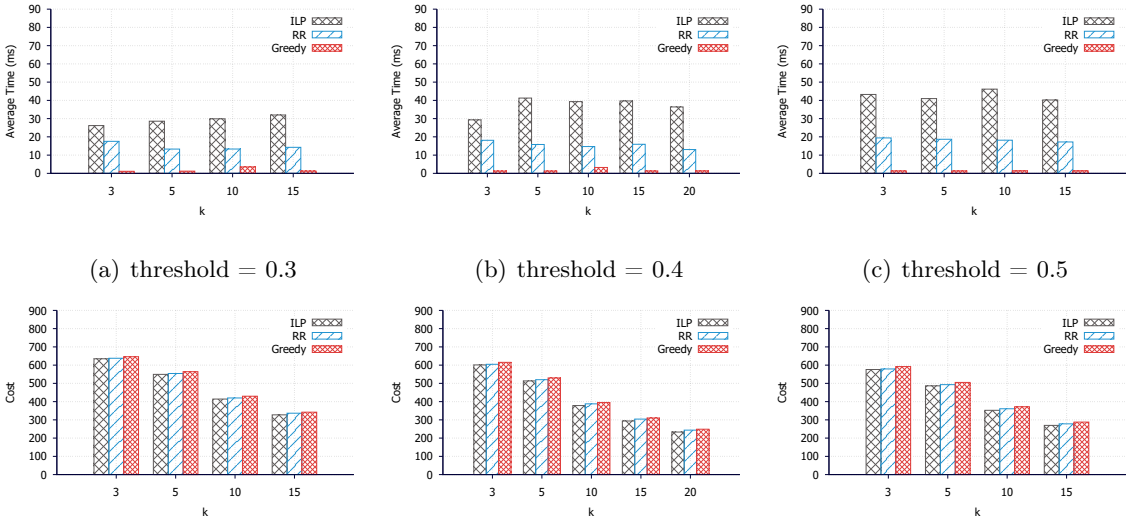
A key observation from these experiments is that Greedy is always the fastest



(a) threshold = 0.3 (b) threshold = 0.4 (c) threshold = 0.5

(d) threshold = 0.3 (e) threshold = 0.4 (f) threshold = 0.5

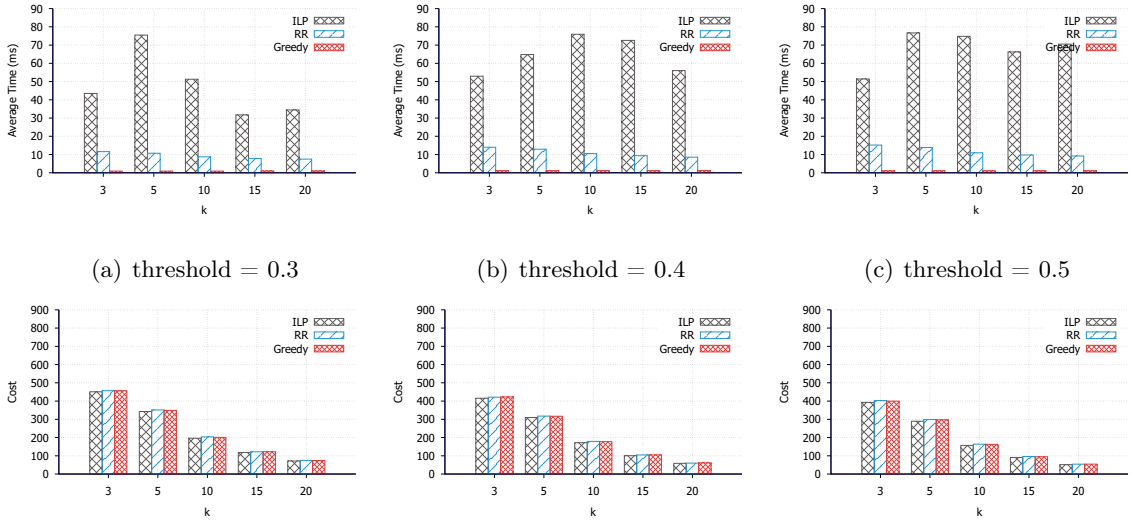
Figure 2.6: Evaluation of Top Pairs on doctor reviews dataset



(a) threshold = 0.3 (b) threshold = 0.4 (c) threshold = 0.5

(d) threshold = 0.3 (e) threshold = 0.4 (f) threshold = 0.5

Figure 2.7: Evaluation of Top Sentences on doctor reviews dataset



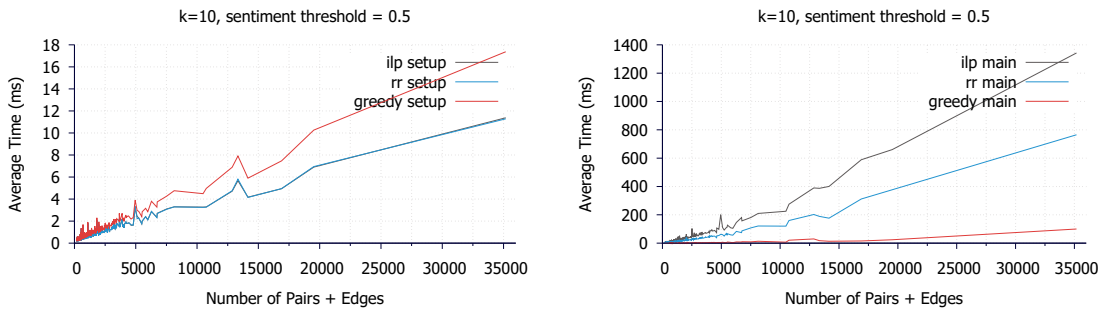
(a) threshold = 0.3 (b) threshold = 0.4 (c) threshold = 0.5

(d) threshold = 0.3 (e) threshold = 0.4 (f) threshold = 0.5

Figure 2.8: Evaluation of Top Reviews on doctor reviews dataset

algorithm while maintaining reasonable costs compared to ILP and RR algorithms. Of course, ILP gives optimal solution and always offers the cheapest cost. The Greedy algorithm has the worst cost but never more than 8% higher than the optimal (within 5% most of time). In terms of time, the Greedy algorithm outperforms ILP by a factor up to 19x, 32x and 63x in the top pairs, top sentences and top reviews problems, respectively. Similarly, Greedy runs faster than RR, at most 14 times, and usually takes only 1–2 ms per doctor. RR algorithm often works similarly to ILP regarding cost, specifically, the difference is about 1-2 percent. The speedup of RR over ILP is usually about 2–5x. This is because RR only solves a Linear Program system and then randomizes the solution instead of finding an optimal integer solution.

We also notice that with the same threshold, the cost decreases from top pairs to top sentences, and then to top reviews problem. The reason is that a sentence or review



(a) setup time

(b) main-process time

Figure 2.9: Algorithms' scalability on doctor reviews dataset

can have multiple pairs, so they typically cover more pairs than a single pair can cover. Therefore, k sentences or reviews usually cover more pairs than k pairs can, which leads to smaller costs. Similarly, the elapsed time of all algorithms for top sentences/reviews problem are larger than for top pairs problem. It's because for top sentences/reviews, there are more connections (edges) between selecting candidates and pairs to consider.

In general, the results suggest that our problem has latent structures friendly to Greedy algorithm. Therefore, the optimal solution from ILP algorithm seem to be close to the one of Greedy algorithm which can be achieved much faster. Because of this reason, we choose Greedy algorithm for the next qualitative experiments.

Algorithm's scalability

In this experiment, we evaluate the algorithms' scalability over various problem sizes, that is, we vary the number of pairs plus the number of edges $(n + m)$. When there are multiple doctors with the same $(m + n)$, we report the average time of these doctors. For brevity, we only present the results for the k-pairs coverage problem, and for the case

of $k = 10$, sentiment threshold $\epsilon = 0.5$. It is also worth noting that for the other cases, algorithms have similar behaviors.

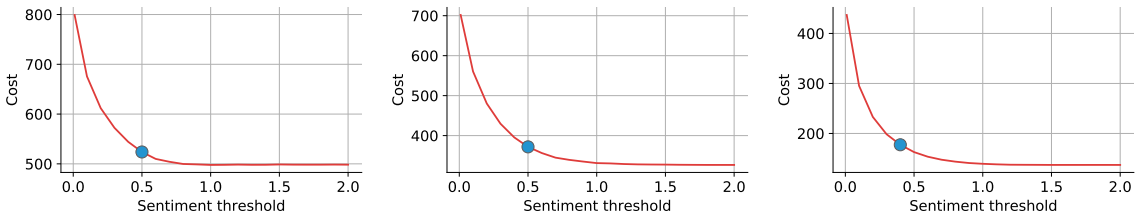
In Section 2.4, we mentioned that the running time of the setup part is roughly linear on $|P| = n$ for our instances. This trend is observed in Figure 2.9(a) since in our sparse graph instances the number of edges (m) is also roughly linear to the number of pairs (n). Note that for the Greedy algorithm we also count the time of initializing max-heap into setup time, thus it is a bit higher than the others. The main-process times are presented in Figure 2.9(b). It's not surprising that Greedy always outperform the others significantly and ILP is the worst candidate. Even though we can get the worst case guarantee for ILP, LP solvers, ILP and Randomized rounding algorithms finish in reasonable time. For the Greedy algorithm the main process is dominated by the setup part. The results suggest that the algorithms can be applied to the real-life situations, especially the Greedy algorithm that not only have the poly-time upper bound in theory but also perform really well in practice.

2.5.3 Qualitative Evaluation

The goal of this section is to study the quality of the summarization achieved by the proposed algorithms, compared to several state-of-the-art baselines. We focus on the sentence selection problem variant, which offers a balance between conciseness and semantic completeness.

Selecting sentiment threshold ϵ used by greedy algorithm:

We first show how we automatically select a sentiment threshold ϵ used by our greedy algorithm. Note that this sentiment threshold is independent of the sentiment thresh-



(a) k-pairs coverage with $k = 10$ (b) k-sentences coverage with $k = 10$ (c) k-reviews coverage with $k = 10$

Figure 2.10: Cost – sentiment threshold trade-off on doctor reviews dataset. Elbow is the blue dot.

olds used in the *coverage measure* during our evaluation (Figures 2.11) to ensure fairness. Specifically, we select the threshold value ϵ for which the rate of covered sentences significantly drops if we further increase ϵ. For that, we employ the elbow method, as shown in Figure 2.10 for $k = 10$ on doctor review dataset, where we observe that there is an L-curve between sentiment threshold and cost. Hence, we choose as sentiment threshold the elbow point of the curve. The sentiment threshold varies with step 0.1 from 0 to 2, as 2 is the maximum possible difference between two sentiment values in [-1, 1]. Theoretically, for this kind of L-curve, the elbow point is where the second derivative of the curve's function is largest. However, since we don't know the curve's function, we use a common trick, where the elbow is the point on the curve with the largest distance to the line connecting the curve's start and end points. In our case, most of the time the elbow is close to sentiment threshold of 0.5. Intuitively, this sentiment threshold is also reasonable in the sense that a very positive sentiment of value 1.0 can cover a positive sentiment of value 0.5. Therefore, we choose sentiment threshold 0.5 for our greedy summarizer.

Baseline summarization methods: We compare our method with popular baselines

on the problem of selecting k sentences from original set of reviews into summaries. Our baselines come from two areas: one from opinion summarization approach, and the other from multi-document summarization.

Specifically, the first baseline method to select top k sentences is adapted from Hu et al. [31]. This algorithm was designed to summarize customer reviews of online shopping products. It first extracts product aspects (attributes like "picture quality" for product "digital camera"), then classifies review sentences that mention these aspects as positive or negative, and finally sums up the number of positive and negative sentences for each aspect. To have a fair comparison, we adapt their method to select top k sentences into summaries. We first count the number of pair (concept, positive) or (concept, negative), for example: aspect "picture quality" with sentiment "positive" occurs in 200 sentences. Then, we select k most popular pairs and return one containing sentence for each selected pair. Note that the aspect extraction task is common in both the baseline and our methods. We refer to this baseline as *"most_popular"* since their summarizer favors the most popular aspects.

The second baseline from the opinion summarization area is adapted from a review summarizer of local services (such as hotels, restaurants) described by Blair-Goldensohn et al. [8]. This method selects the (aspect, positive/negative) pairs proportionally to the pair's frequency instead of selecting the most popular pair as in *"most_popular"* method. Then, it pick the new, most extremely polarized sentence to represent each selected pair (concept, positive/negative). In our experiments, we name this summarizer as *"proportional"*.

The other set of baselines are popular extractive multi-document summarizers that are agnostic to a concept's sentiment orientation. Contrasting to abstractive summarizers

Table 2.2: Baseline summarizers

Most_popular	[31]	pick representative sentences of popular aspect-polarity pairs
Proportional	[8]	pick representative sentences with extreme sentiments after selecting aspects proportionally
TextRank	[52]	no sentiment, use sentence graph with word overlap for sentence similarity
LexRank	[24]	no sentiment, use sentence graph with cosine-based sentence similarity
LSA-based	[72]	no sentiment, utilize SVD on term-sentence matrix

that compose summaries by creating brand-new sentences, extractive summarizers make use of original documents' sentences, hence it is appropriate to be compared with our method. TextRank [52] summarizer applies PageRank algorithm on text by modelling text as graph of sentences in which sentences' similarity is considered as sentence-to-sentence edge weight. LexRank [24] is another document summarizer relying on a sentence graph for detecting the most important sentences. LexRank is normally incorporated with a reranker such as Cross-Sentence Informational Subsumption [63] to avoid sentence redundancy in multi-document summarization. The last baseline in this line is Latent Topic Modelling (LSA) based summarizer [72], which utilizes the sentence's vector representation calculated using Singular Value Decomposition (SVD) on a term-sentence matrix. In our experiments, We utilize Sumy [7] library for these three methods. We summarize all baselines with brief descriptions in Table 2.2.

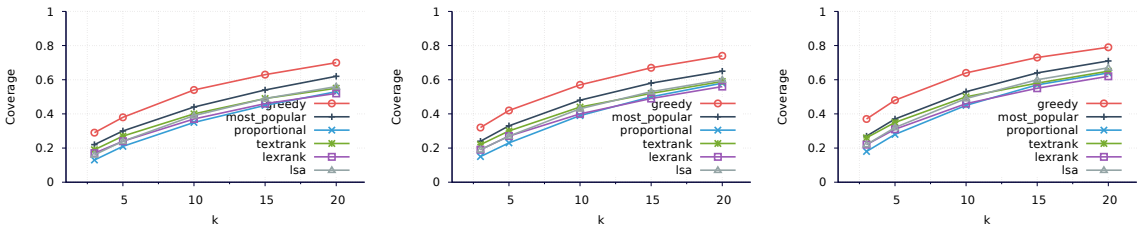
Summary Quality Measures and Results

Coverage measure: We first compare our method (*greedy*) with the baselines based on an intuitive *coverage measure* that is different from the one proposed in our Problem Definition section, to avoid giving an unfair advantage to our method. Specifically, the measure is

defined as the percentage of concept-sentiment pairs covered by that selected k sentences divided by the total number of pairs of a doctor. A sentence *covers* a pair p if the sentence contains at least one pair that covers p. A pair is said to cover another if their sentiment difference is less than or equal to a (sentiment) threshold and their dissimilarity is at most a (distance) threshold. In our experiment we use the path length between concepts of pairs in the ontology as dissimilarity measure. Note that this measure does not consider the successor-descendant relationships between concepts.

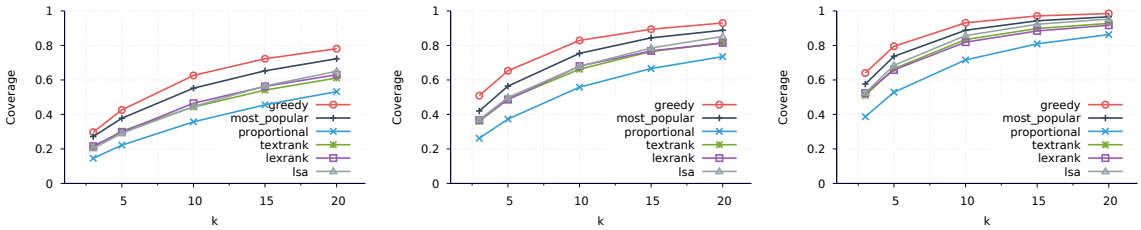
Results: We evaluated this coverage measure for several distance and sentiment thresholds, but only show the results for sentiment threshold of 0.3, and distance threshold of 2, 3 and 4 due to the space limitation, in Figure 2.11 and 2.12. Consistently in all cases, our method outperforms the best performing baselines about 10 – 30% per case. We also notice that the coverage slightly increase from Figure 2.11(a) to Figure 2.11(b) and Figure 2.11(c) (similarly for Figure 2.12) because when distance threshold increases from 2 to 3 and 4, there are more chances that a sentence cover pairs in original reviews. For the same configuration, all methods achieve higher coverage on cell phone reviews dataset than on doctor reviews dataset. This is because aspect hierarchy in cell phone reviews is much narrower and more shallow than medical concept hierarchy, thus it is easier to cover most of sentiment-concept pairs in cell phone review dataset.

Sentiment Error: The second measure, which we refer as *"sentiment error"*, is totally different. The key idea is to look at the difference between every concept's sentiment in the original reviews and that concept's sentiment (extrapolated if concept not in summary) in the summary. That is, for each pair in the original reviews, we find the closest concept in



(a) Distance threshold 2 (b) Distance threshold 3 (c) Distance threshold 4

Figure 2.11: Comparison of coverage measure with sentiment threshold 0.3 on doctor reviews dataset (higher coverage is better)



(a) Distance threshold 2 (b) Distance threshold 3 (c) Distance threshold 4

Figure 2.12: Comparison of coverage measure with sentiment threshold 0.3 on cell phone reviews dataset (higher coverage is better)

the summary and measure the *sentiment distance* between them. In contrast, in Definition 2, we measure the *concept distance* (in hierarchy edges) between a review concept and its nearest covering summary concept.

Recall that we summarize a set of concept-sentiment pairs P by a subset F contained in k sentences. We define the sentiment error of F with respect to P in a root-mean-square error manner:

$$sentiment\_error(P, F) = \sqrt{\frac{1}{|P|} \sum_{p \in P} err_{p,F}^2}$$

where p is a pair of (concept c_p, sentiment s_p). $err_{p,F}$ is the smallest difference between s_p and that concept's sentiments in a pair in F. When concept c_p does not appear in the summary F, we use the sentiments of c_p's lowest ancestor in F if available. When neither c_p nor its ancestors appear in F, we consider a neutral sentiment of 0.

$$err_{p,F} = \begin{cases} \min\limits_{f \in F, c_f = c_p} |s_f - s_p| & : c_p \in F \\ \min\limits_{f \in F, c_f = c_p\text{'s ancestor}} |s_f - s_p| & : c_p \notin F \wedge \\ & \quad c_p\text{'s ancestor} \in F \\ |0 - s_p| = |s_p| & : \text{otherwise} \end{cases} \tag{2.1}$$

The intuition is that the error models the difference of every concept's sentiment and the closest sentiment of that concept or its ancestors in summary. The lower error value means a higher summary quality.

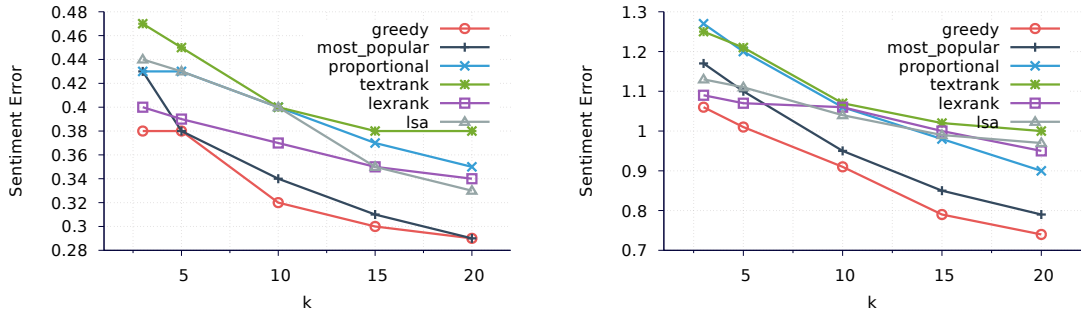
Another version of this measure penalizes the case of missing concept c_p and its ancestor in summary F by considering the largest possible error of c_p's sentiment. In another words, the third branch of Equation (2.1) becomes $err_{p,F} = \max(|1 - s_p|, |-1 - s_p|)$. Note

that $+1$ and -1 are the extreme sentiments in our model. We name this measure version as *"sentiment-error-with-penalization"*.

Results: Figure 2.13 and 2.14 compare the errors of our method and the baselines. On the first measure, "sentiment-error" (Figure 2.13(a) and 2.14(a)), we find that our method always leads to the smallest sentiment error, i.e. highest-quality summaries. It can reduce the error of the second best performance method ("most_popular") on doctor reviews dataset by 4.2% on average, and other methods by 15% on average. Similar improvement numbers on cell phone reviews dataset are 4.1% and 14.6% on average respectively.

The multi-document summarization methods generally perform poorly since they ignore the sentiment. Our method reduces those multi-document summarizers' error by up to 23.7%. The errors of all methods drop when the number of summary sentences increases, as expected.

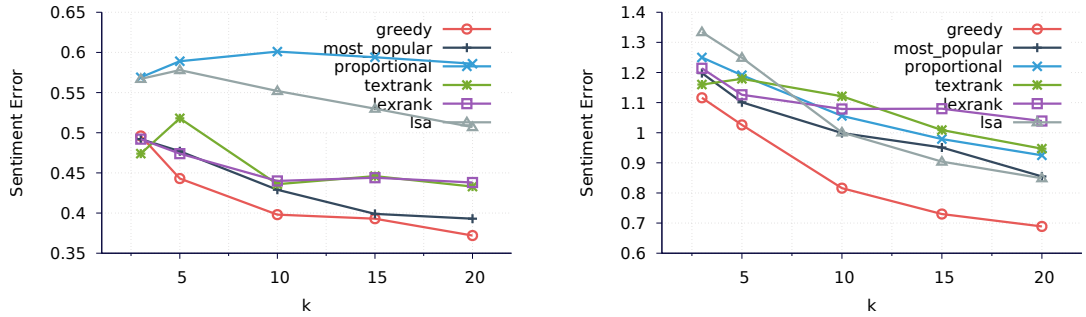
On the "sentiment-error-with-penalization" measure (Figure 2.13(b) and 2.14(b)), our method beats all baselines with larger margins. Specifically, our method improves the error of second best performance method ("most_popular") on doctor reviews dataset by 7%, and other methods by 15.8% on average. Those numbers on cell phone reviews dataset are 14.9% and 19.8% respectively. This result indicates that missing concepts in summary problem is more severe in baseline methods, and our method is smarter in choosing sentences.



(a) Without penalizing missing concepts

(b) With penalizing missing concepts

Figure 2.13: Comparison of sentiment error on doctor reviews dataset (lower error is better)



(a) Without penalizing missing concepts

(b) With penalizing missing concepts

Figure 2.14: Comparison of sentiment error on cell phone reviews dataset (lower error is better)

2.6 Related Work

Multi-document Summarization: This problem has been studied for two decades; the most well-known applications are summarizing online news articles [18]. Goldstein et al. presented a typical method [27], which extend single-document summarization techniques. A key difference is that there is more redundancy across documents of a similar topic than within a single document. This is an observation we also adopt in our work. In short, the proposed method [27] constantly select a passage (normally sentence) that has the high "marginal relevance" score (combination of relevance and novelty) into the summary, and finally re-order candidates to maintain content cohesion. The other approaches based on cluster extraction such as MEAD summarizer [63] first extracts common topics/clusters/-centroids using respectively linguistic features or word statistics from input documents, then selects one sentence per cluster into summary. TexRank [52] and LexRank [24] are two popular, similar methods based on building weighted graph of document sentences, which are rated by Pagerank algorithm to pick the important ones. Lastly, Steinberger et al. [72] proposed an LSA-based summarizer that utilizes sentence's vector representation in their latent index space. However, none of above methods consider the sentiment in input documents. We incorporate some of these methods (TextRank, LexRank and LSA-based) as baselines in our evaluations (Section 2.5).

Sentiment Analysis: The methods fall into two categories, using unsupervised or supervised learning. The unsupervised methods [80, 74] focus on building a comprehensive opinion word dictionary, or use linguistic rules to find opinion phrases containing adjectives or adverbs in a document. An early supervised learning method [59] applies a Bag-Of-Word

model to classify movie reviews as positive or negative. Recently, a common approach [44] is to use neural network model to extract the better review's vectors, thus get the better results on sentiment classification task. Any of these methods can be plugged into our framework.

Aspect Extraction: Recent research focuses on online review analysis, social media and online shopping. A common task is to extract the product aspects. Traditional methods [31, 62] use association mining to find frequent aspects, then apply pruning rule to remove meaningless, redundant ones; later they also have a rule to discover additional infrequent aspects based on both frequent ones and opinion words. A more advanced technique [61] estimates *Point-wise Mutual Information* (PMI) score between phrases using their Web search engine hit counts to find phrases relation. For example, they extract noun phrases in reviews, then evaluate each one by calculating their PMI score with product relation phrases such as "of scanner", "scanner has" and "scanner comes with" for Scanner class. Jakob and Gurevych [33] adapted Conditional Random Field (CRF) technique to solve this task as an information extraction problem. A semi-supervised approach based on topic modelling extract product aspects as multi-grain topics [76, 57]. Extracting aspects is outside the scope of this work. We use Metamap [4] in our experiments to extract medical aspects (concepts).

Opinion Summarization: The most popular approach is based on aspect extraction. Typical methods [31, 22] first extract product aspects from online customer reviews, then classify containing sentences as positive or negative, and report the number of positive/negative sentences for each aspect. Different ways of presenting opinion summaries have been proposed; for example, showing aggregated rating along with represen-

tative phrases [50], or sentences [8] for each aspect. Different from this kind of statistical summaries, Tsaparas et al. [79] and later Lappas et al. [42] formulate the problem as selecting k reviews that optimize the aspect coverage while rewarding high-quality reviews, or maintaining their proportion of aspect opinions. Instead of sentences/reviews selection approaches, Carenini et al. [12] propose to generate summaries using product aspects and a set of templates varying on the aspect's sentiment. These templates are prepared beforehand with text passages not included in the original documents.

A key difference of this work from all the above works is that they do not consider the relationships between the aspects nor a continuous sentiment scale.

2.7 Conclusions

We introduced a novel review summarization problem that considers both the ontological relationships between the review concepts and their sentiments. We described methods for extracting concepts and estimating their sentiment. We proved that the summarization problem is NP-hard even when the concept ontology is a DAG, and for that we presented efficient approximation algorithms. We evaluated the proposed methods extensively with both quantitative and qualitative experiments. We found that the Greedy algorithm can achieve quality comparable to the optimal is much shorter time, comparing to other algorithms. Moreover, using various coverage measures and sentiment error measures, we show that the Greedy outperforms a baseline method on selecting k sentences to summarize real reviews.

Chapter 3

Decrease Product Rating Uncertainty Through Focused Reviews Solicitation

3.1 Introduction

Product reviews are essential in e-commerce to alleviate the lack of direct physical contact with the products. Specifically, online reviews have been shown to affect a product's uncertainty, which is crucial to e-customers' shopping decision [21]. Kim and Krishnan [40] noted that consumers are unlikely to buy expensive products (defined as higher than $50) online if there is a high degree of product uncertainty, even if they have a lot of online shopping experience. There are several approaches that e-commerce companies have utilized to mitigate this product uncertainty issue such as providing detailed descriptions, including multimedia and virtual reality tools. Most notably, soliciting customer reviews has become a standard of modern online shopping.

In this chapter, we focus on studying review solicitation strategies that maximize the effect of reviews to the decrease in product uncertainty. Khare et al. [37] found that reviews' volume and the level of consensus have a fundamental impact on consumer judgment. Hence, an effective review solicitation strategy must account for both these factors.

Existing e-commerce platforms typically ask users to write a free-text review. These reviews can then be analyzed by feature and sentiment extraction methods (Section 3.2) to estimate the overall opinion of reviewers for each aspect (feature) of a product. Other websites provide a static (predefined) set of aspects for the user to rate, typically with a score from 1 to 5. For example, "How clean was your room?" or "How would you rate the reliability of the car?"

A key drawback of existing review solicitation methods is that some aspects receive too many ratings, which is especially wasteful if reviewers generally agree with each other. For example, consider a product "smartphone" with aspects "screen," "battery," "design" and so on. Hundreds of reviewers may rate the "screen" as 5-stars. Conversely, a more controversial aspect, e.g. the "speed," may only receive a few ratings. This leads to a review profile with high uncertainty, as users typically try to compare various products across several aspects (features) by using past users' reviews.

An example of a smartphone's review profile is presented in Table 3.1, which intuitively shows that screen has high rating with high confidence, battery has low rating with high confidence, and speed has high rating with low confidence. A key question that this chapter studies is: *given the current reviews profile of a product, is it better to let users write a free-text review (and then extract the aspects and opinion using existing meth-*

Table 3.1: A review profile; numbers are rating counts.

	⋆	⋆⋆	⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆⋆
Screen	0	0	0	5	31
Battery	26	10	3	0	0
Speed	0	1	2	0	3

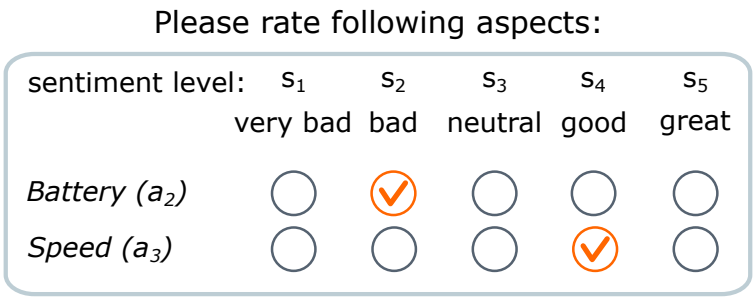


Figure 3.1: Ask a customer about a smartphone

ods [31, 62, 61, 33, 77, 57]), or to ask the user to rate a small number of carefully selected aspects as in Figure 3.1? A second question is: *how should this small set of aspects be selected, given the current review profile?*

We study these two questions in a principled manner by first considering a Bayesian statistical model to estimate the probability distribution of each aspect's rating and then dynamically selecting aspects whose estimated ratings have the highest posterior variance. Intuitively, this method solicits reviews for the aspects that have few reviews or have diverse opinions. This means subsequent users may be asked to rate different aspects of the product.

We understand that reviews' uncertainty may also be affected by other factors like spam reviews [82, 34], or the helpfulness of the text of the reviews [56, 41, 6]. These are important factors, orthogonal to our focus, and outside the scope of this work.

To design and compare various aspect selection methods, we must first come up

with a reasonable definition for review profile uncertainty, as no such standard measure exists in the literature. In our method, we estimate a review profile uncertainty by the expected rating variance of each aspect, which we model based on a well-accepted Bayesian inference model. This model is consistent with the aforementioned points that reviews' volume and consensus are the key factors in consumer's evaluation of a product, as a high number of reviews or high review agreement reduce a rating's posterior variance. To avoid comparing various review solicitation methods based solely on the variance of the aspects, which may favor our proposed methods, we also consider other uncertainty measures based on the confidence interval (from a frequentist statistician's point of view, in contrast to our Bayesian measure), and the number of aspects whose confidence is above a threshold.

Besides a head-to-head comparison between active and passive review solicitation methods, we propose a third approach that leverages the best of both methods. In particular, this approach deploys a traditional free-form text reviewing interface first, then dynamically selects a small set of unseen aspects with high rating uncertainty to ask users to rate. In this manner, we preserve the rich context of text reviews, while exploiting the rating uncertainty reduction capability of active solicitation method. We also show that the extra cost of asking additional aspects is negligible compared to the pure passive solicitation's cost in terms of user spent time.

We next extend our methods to account for dependencies among a product's aspects. For example, if "screen" and "contrast" are two correlated aspects and there are many and in-agreement reviews for "screen," it may be wasteful to solicit reviews for "contrast." For this, we consider a dependency-aware Bayesian inference model to estimate the correla-

tion of two aspects. Then, we generalize the previous definition of expected rating variance to infer an aspect's variance from others if they are highly correlated.

We compare our methods on two real datasets: Amazon reviews with annotated aspect ratings introduced by Bing Liu, et al. [31, 22] and crawled automobile reviews from `edmunds.com`. We first compare our method to the passive text-based solicitation method, which is simulated by picking top aspects based on the order in which they appear in reviews. Users' answers are reproduced using the actual aspects' sentiments extracted from their free-text reviews. In another group of experiments, we compare our method to various baselines that also select set of aspects to solicit users. In these cases, we utilize random generators to generate sentiments as the answers. We also experiment with the realistic situation that a user responds to a question with a given aspect-specific probability, instead of assuming that the user always rates an aspect. That is, sometimes the user skips the question or responds "I don't know." In the last group of experiments, we compare our hybrid reviewing interface to pure passive solicitation with consideration of user effort cost. We consider three uncertainty measures: rating variance (as introduced in our model), rating confidence interval length, and ratio of highly confident aspects (independent of our model). Our contributions are summarized as follows:

- We define the problem of dynamically selecting aspects to solicit targeted reviews to reduce uncertainty and propose a principled method for that based on canonical Bayesian inference (Section 3.3).

- We propose a hybrid method that takes advantages of both active and passive review solicitations (Section 3.4).

- We extend the problem in two ways. First, we consider the practical case that users do not always respond to a question (Section 3.5.1). Second, we propose an extension of our aspect selection method that considers aspects' correlation (Section 3.5.2).

- We conduct detailed experiments (Section 3.6) on two real-world datasets, which show that our methods lead to superior review profiles compared to passive text-based solicitation and other aspect selection methods, with or without the consideration of user response probability. We also show that our hybrid reviewing interface significantly improves the reviews uncertainty compared to the passive solicitation method, with little extra user cost (time).

- We published our code and used datasets on our supporting web page [43].

The remainder of the chapter is organized as follows: we discuss the related work in Section 3.2, and the conclusions and future work in Section 3.7.

3.2 Related Work

Commercial Reviewing Web Sites: Most sites solicit free-text reviews, along with an "overall rating" typically expressed with 1 to 5 stars. Other web sites have a small, predefined set of questions that they ask reviewers; for instance, `vitals.com`, which is a doctor reviewing site, asks users to assign a score to "bedside manner" and "courteous staff." The only web site that we found that has a dynamic set of questions is `tripadvisor.com`, which asks users to rate different hotel aspects (e.g., "service," "location" and "sleep quality") for different hotels. However, we have no knowledge of how these aspects are selected as

this is a proprietary system.

Dynamic Questionnaires: USHER [14] is a system for form-based survey design that aims to improve the quality of collected data. USHER uses a probabilistic model on the form questions, learned from previous form submissions, to adapt the form layout (question ordering) dynamically to emphasize the most important questions, or re-ask questions that may have been answered incorrectly. A key difference is that in USHER the goal is to collect information about all the questions from each user, whereas our goal is to collect enough (and reliable) information for each product aspect. For this, we analyze our aspect ratings' certainty, which is not the case in USHER.

Multi-armed Bandit Problem: This is one of the fundamental problems in Artificial Intelligence [5]. In its simple form, a gambler presented with a row of slot machine must decide her playing strategy, i.e. which machine to play next given the sequence of past plays, to maximize her reward. The key property of this problem is that rewards of successive plays on a machine i are independent and identically follow a distribution of an unknown expected value R_i. In our problem, the reward is the decrease in the uncertainty of each aspect, where these uncertainties may be dependent to each other (Section 3.5). Another difference is that in the multi-armed Bandit problem, the gambler is guaranteed the highest reward in the long run if she found the machine with the highest expected reward value, then played on that machine only. In our case, there is no aspect that will forever produce highest expected uncertainty drop when we keep getting more rating of this aspect.

Reviews Analysis: There has been much work on analyzing text reviews. These works generally have two phases. First, they extract aspects (features) like "zoom," and second,

they estimate the sentiment associated with each aspect using its surrounding context. *These works are complementary to our work, as they facilitate converting text reviews to structured review profiles, which can then be processed by our algorithm to select which aspects to elicit in future reviews.*

Aspect Extraction: The most common approaches to extract aspects from product reviews are based on keyword statistics and syntactic rules. Existing works [31, 62] use association rule mining to find frequent aspects, and then filter out meaningless or redundant ones using predefined syntactic dependency-based rules. After that, these frequent aspects and opinion words are utilized to discover more infrequent aspects using another set of rules. Another technique [61] decides if an aspect candidate is actually an aspect by checking the *Point-wise Mutual Information* (PMI) score between it and its product class using their Web search engine hit counts. Another approach, adopted by Jakob and Gurevych [33], models this task as an information extraction problem and applies conditional random field techniques to extract aspects. Topic modelling has also been used for this problem, as in Titov and McDonald [77], who discover global and local aspects; and Mukherjee and Liu [57], who extract and categorize aspects given some seeds.

Sentiment Analysis: This problem has been investigated extensively, and has been comprehensively surveyed by Liu and Zhang [49]. Traditional methods [74] focus on creating a comprehensive, good dictionary of opinion words that are looked up when analyzing text reviews. Other authors such as Turney [80] exploit syntactic patterns to detect opinion phrases containing adjectives or adverbs. A supervised learning algorithm was first introduced to classify movie reviews as positive or negative based on vectors of reviews using the Bag-

of-Words model [59]. In this model, authors experimented with Naive Bayesian and SVM classifiers that offer accurate results. Recently, the use of deep neural networks and representation learning have improved the performance of this task significantly [51, 54, 17, 20, 44]. For instance, Le, et al. [44] use an unsupervised neural network model to learn reviews' representational vectors that are later fed to a standard supervised classifier for sentiment analysis.

3.3 Modeling a Product's Review Profile and Aspect Selection Algorithm

3.3.1 Problem Definitions

An online product (or service) has a set of aspects (also referred as attributes or features in other papers) denoted as $a_1, a_2, \ldots, a_m$. Each aspect receives ratings from l sentiment (star) levels $s_1, s_2, \ldots, s_l$.

The review profile of a product is a summary of the aspect ratings, as exemplified in Table 3.1. To model the quality of the review profile, we define the *review profile's statistical summary (RPSS)* as a set of tuples:

$$< a_h, r^{a_h}, cert^{a_h} > \qquad \text{with } h = 1, \ldots, m \tag{3.1}$$

where r^{a_h} is the expected rating of a_h and $cert^{a_h}$ is the certainty level of r^{a_h} estimation, which are discussed in Section 3.3.2. We also call $uncert^{a_h}$ as the uncertainty level inversely proportional to $cert^{a_h}$ (i.e. $uncert^{a_h} = 1/cert^{a_h}$). A particular aspect a_h gets $n_i^{a_h}$ votes for sentiment s_i $(i = 1, 2, \ldots, l)$, and in total n^{a_h} ratings $(n^{a_h} = \sum_i n_i)$.

This chapter studies the problem of selecting the *top-k Uncertain Aspects (k-UA)*: *Given current users rating history:* $< a_h, n_i^{a_h} > (h = 1, \ldots, m; i = 1, \ldots, l)$*, which are the k aspects to ask the next reviewer to rate in order to optimize the review profile?* In Section 3.5 we extend this problem definition to consider aspect rating correlations, by accounting for the co-occurrences of aspect ratings within reviews.

Note that the top-k aspects are recomputed for each new reviewer. The computational cost is negligible, so even for high throughput of reviews, the algorithm can dynamically update the top-k aspects. The product's aspects can either be explicitly listed at the reviewing web site, or may be extracted automatically from a text review. In the former case, the reviewer selects a number of stars for each aspect, and in the latter case the sentiment is estimated automatically. These methods are discussed in detail in Section 3.2.

In the following sections, we will present our Bayesian approach to model an aspect's certainty level in a RPSS, and then propose our algorithm for the *k-UA* problem.

3.3.2 Bayesian Inference Model

Our model focuses on measuring aspect a_h's uncertainty level $uncert^{a_h}$ of its expected rating r^{a_h}. In this section, to simplify the notation we ignore the superscript a_h in $r^{a_h}, uncert^{a_h}$ and $n_i^{a_h}$. Let $p = (p_1, p_2, \ldots, p_l)$ be a random vector representing the probabilities (degree of belief) that users rate the aspect with $s_1, s_2, \ldots, s_l$ stars, respectively. We follow a typical Bayesian inference for categorical data [1] to account for this probability vector. In particular, each sentiment level s_i is a category that users' ratings fall in.

Suppose that the prior distribution of $p = (p_1, p_2, \ldots, p_l)$ is a Dirichlet distribution of order $l \geq 2$ with parameters $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_l)$, $\alpha_i > 0, \forall i$: $g(p) = \frac{1}{B(\alpha)} \prod_{i=1}^{l} p_i^{\alpha_i - 1}$,

where $B(\alpha)$ is the Beta function. It is common to consider the uniform case as the prior: $\alpha_i = 1$ $(\forall i)$ since the likelihood will dominate the prior over time.

Also assume that the likelihood $f(n|p)$ of observed data $n = (n_1, \ldots, n_l)$ (sentiment counts) is a multinomial distribution: $f(n|p) = \frac{N!}{n_1! \ldots n_l!} \prod_{i=1}^{l} p_i^{n_i}$, where $N = \sum_{i=1}^{l} n_i$ is the total number of sentiment counts. Hence we have the posterior:

$$h(p|n) \propto f(n|p)g(p) = \frac{N!}{n_1! \ldots n_l!} \times \frac{1}{B(\alpha)} \times \prod_{i=1}^{l} p_i^{n_i+\alpha_i-1}.$$

Let $\beta_i = n_i + \alpha_i$, $\beta_0 = \sum_i \beta_i = N + \sum_i \alpha_i$. Then the posterior $h(p|n)$ is also a Dirichlet distribution with parameter $(n_1 + \alpha_1, \ldots, n_l + \alpha_l)$, or $(\beta_1, \ldots, \beta_l)$ with mean, variance, and covariance, respectively:

$$E[p_i|n] = \frac{n_i + \alpha_i}{\Sigma_{i=1}^{l}(n_i + \alpha_i)} = \frac{\beta_i}{\beta_0}$$

$$Var[p_i|n] = \frac{\beta_i(\beta_0 - \beta_i)}{\beta_0^2(\beta_0 + 1)} \tag{3.2}$$

$$Cov[p_i, p_j|n] = \frac{-\beta_i\beta_j}{\beta_0^2(\beta_0 + 1)} \qquad \text{for } i \neq j \tag{3.3}$$

The aspect's expected rating is $r = \sum_i s_i p_i$, and hence

$$E[r|n] = E[\sum_i s_i p_i|n] = \sum_i s_i E[p_i|n] = \sum_i s_i \frac{\beta_i}{\beta_0}$$

$$\begin{aligned} Var[r|n] &= Var[\sum_i s_i p_i|n] \\ &= \sum_i s_i^2 Var(p_i|n) + \sum_{i \neq j} s_i s_j Cov(p_i, p_j|n) \\ &= \frac{1}{\beta_0^2(\beta_0 + 1)}[\sum_i s_i^2 \beta_i \beta_0 - \sum_i \sum_j s_i s_j \beta_i \beta_j] \end{aligned} \tag{3.4}$$

Since $Var[r|n]$ reflects the fluctuation of an aspect's rating around its expected value, $Var[r|n]$ can be interpreted as the uncertainty measurement of our estimation of the aspect's rating, i.e. $uncert = Var[r|n]$. Variance $Var[r|n]$ also has an intuitive property that it is roughly inversely proportional to the number of votes N (via β_0 in the denominator of Equation (3.4)). If an aspect has a very high uncertainty value, i.e. $Var[r|n]$, it means that we are not ready to make a conclusive estimation of its rating. Also note that, asking a controversial aspect still alleviates its variance slowly even if its new ratings are truly polarized. In the common practice, a uniform prior is used in this Bayesian inference, thus $\alpha_i = 1$. As a result, $\beta_i = n_i + 1$ and $\beta_0 = N + l$. Note that in our experiments we also consider alternative measures of uncertainty when comparing the proposed algorithms.

3.3.3 Aspect Selection Algorithm

We present our solution to the *k-UA* problem in Algorithm 3. In particular, Lines 2-3 set up common uniform prior parameters, while lines 5-7 compute posterior parameters for every aspect. We finally calculate rating variance of all aspects in line 8, then output the top k aspects with highest variances (i.e., degree of uncertainty).

Note that $Var[r|n]$ can be computed faster using vectorized version of Equation (3.4). Specifically, $Var[r|n]$ is the variance of a linear combination of column vector s and random vector p, so $Var[r|n] = s^T \Sigma s$, where Σ is the covariance matrix built up using Equations (3.2) and (3.3) that can be vectorized as well.

Algorithm 3 Highest variance pick

Input: previous vote counts $n_1, \ldots, n_l$ of aspects, number k

Output: k aspects

1: **procedure** PICK_HIGHEST_VARIANCE

2: **for all** i in $1 \ldots l$ **do**

3: $\alpha_i = 1$ ▷ uniform prior for every aspect

4: **for all** aspect a **do**

5: **for all** i in $1 \ldots l$ **do**

6: $\beta_i^a = n_i^a + \alpha_i$ ▷ posterior parameters

7: $\beta_0^a = \sum_{i=1}^{l} \beta_i^a$

8: Calculate $Var[r^a|n^a]$ using Equation (3.4)
return top k aspects with highest $Var[r^a|n^a]$

3.3.4 Toy Example

To explain the intuition of our model, consider a toy example where we are looking at a smartphone with four aspects: weight, cost, battery and design. Each aspect can be rated with 1, 2 or 3 stars (i.e., bad, neutral or good). The previous ratings of these aspects are presented in Table 3.2. The question is which aspects we should ask users about to improve this smartphone's RPSS? Following the previous model, we can model aspects' expected rating as Dirichlet posteriors that are demonstrated in Figure 3.2 and the RPSS in Table 3.3. We then use Algorithm 3 to calculate each aspect's rating variance and order them to select the k most uncertain aspects. In this case, the algorithm will pick aspect "Design" first, then "Battery," "Cost" and finally "Weight." Design is a clear choice since it has far fewer ratings to estimate its rating with high confidence. The other three aspects

Table 3.2: Toy example of 4 aspects with counts of 1, 2 or 3 stars respectively.

	Weight	Cost	Battery	Design
Star count	0, 5, 28	4, 9, 20	11, 11, 11	1, 3, 7

Table 3.3: RPSS of Table 3.2, where *uncert*=variance.

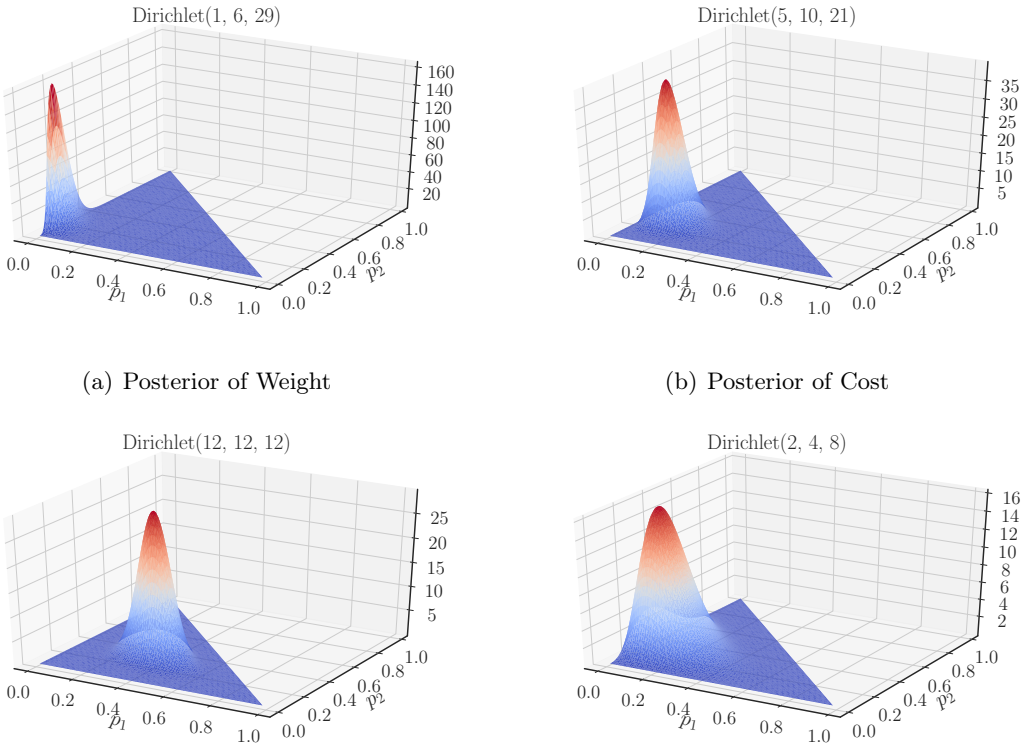
	Weight	Cost	Battery	Design
Expected Rating	2.78	2.44	2	2.43
Variance	0.006	0.014	0.018	0.035

have the same number of ratings but Battery has more diverse opinions, so it is selected next. Weight is picked last because of its high rating count and very skewed rating distribution.

3.4 Hybrid Reviewing Interface

In this section, we propose an intuitive way to combine both the active solicitation method (i.e. aspect selection algorithm in Section 3.3.3), and the passive (free-form text) review solicitation into a hybrid reviewing interface. Specifically, users (reviewers) start with a traditional text form for writing review. Right after they finish their text review, the system applies a revised version of Algorithm 3 to select a few un-reviewed aspects to ask users to rate.

Figure 3.3 depicts this process, where a mobile user first writes his/her review on a screen (on the left), then continues to another screen (on the right) with the specific aspect rating questions after hitting the "next" button. We formalize the modified aspect selection algorithm in Algorithm 4. Note that this algorithm only selects aspects which are not mentioned in the current user's text review. Since users are asked for additional



(a) Posterior of Weight

(b) Posterior of Cost

(c) Posterior of Battery

(d) Posterior of Design

Figure 3.2: Toy example: posteriors of aspects' rating

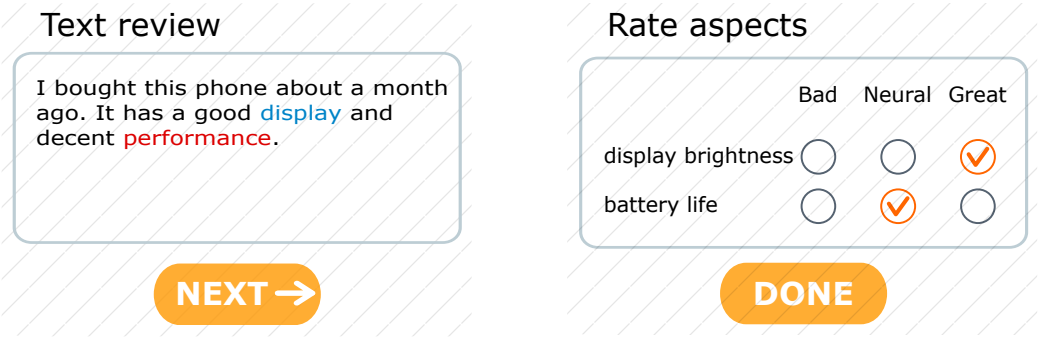


Figure 3.3: Example of a hybrid reviewing interface

questions after finishing their text reviews, there is extra burden on the user side. We rely on previous work on typing speed and questionnaire response time to estimate this extra user time spent.

3.5 Extensions for Response Probability and Aspects Correlation

3.5.1 Account for Response Probability

So far, we assumed that the user always rates an aspect when asked. In reality, this is not true due to numerous reasons. Sometimes, users may be lazy to answer, or may not be confident, or have enough information about the requested aspects such as the phone's durability, car's safety features. This suggests that the likelihood of user response is aspect-specific. We refer to this as *user response probability*.

Algorithm 4 Unseen, highest variance pick

Input: previous vote counts $n_1, \ldots, n_l$ of aspects, number k, current text review R

Output: k aspects

1: **procedure** PICK_HIGHEST_VARIANCE_UNSEEN

2: $UnSeen = \emptyset$

3: **for all** i in $1 \ldots l$ **do**

4: $\alpha_i = 1$ ▷ uniform prior for every aspect

5: **if** aspect $a_i \notin R$ **then**

6: $UnSeen$.add(a_i)

7: **for all** aspect a **do**

8: **for all** i in $1 \ldots l$ **do**

9: $\beta_i^a = n_i^a + \alpha_i$ ▷ posterior parameters

10: $\beta_0^a = \sum_{i=1}^{l} \beta_i^a$

11: Calculate $Var[r^a|n^a]$ using Equation (3.4)

return top k aspects in $UnSeen$ with highest $Var[r^a|n^a]$

Table 3.4: Counting when two aspects were rated together by an user.

	Design–1	Design–2	Design–3	
Cost–1	3 (n_{11}, p_{11})	2 (n_{12}, p_{12})	0 (n_{13}, p_{13})	5 (n_1^c)
Cost–2	1 (n_{21}, p_{21})	5 (n_{22}, p_{22})	2 (n_{23}, p_{23})	8 (n_2^c)
Cost–3	1 (n_{31}, p_{31})	4 (n_{32}, p_{32})	7 (n_{33}, p_{33})	12 (n_3^c)
	5 (n_1^d)	11 (n_2^d)	9 (n_3^d)	

In our experiments, we estimate it using the following forumla:

$$response\_prob_{aspect\ a} = \frac{number\_of\_reviews(a)}{total\_number\_of\_reviews} \tag{3.5}$$

That is, we normalize the number of reviews containing an aspect by the total number of reviews in the entire dataset. Even though this equation does not reflect all factors regarding to an aspect response probability, it is reasonable enough for our dataset.

3.5.2 Account for Correlation between Aspects

Section 3.3 provides a framework to model the uncertainty level of aspect ratings, where aspect ratings are assumed to be independent of each other. However, in practice aspects are often correlated. For example, screen and brightness, or design and easy-to-use are similar to each other, and often receive similar rating. Intuitively, if one of two highly correlated aspects (e.g., "screen") has high rating certainty, then it is less important to solicit more ratings for the other aspect (e.g., "brightness"). Next, we first show how to estimate the correlation between the ratings of two aspects, and then show how this can be used to define a correlation-aware version of the uncertainty score of each aspect (recall that the aspect selection algorithm selects the k aspects with the highest uncertainty).

To estimate the correlation of two aspects, we propose to look at their ratings at the same time. In particular, we count the number of times that two aspects were rated

together in the same review. For instance, in Table 3.4 we consider two aspects (cost and design) in a three-star system. Using similar notation as before, n_{ij} and p_{ij} are, respectively, the number of reviews and the probability users rate aspect "cost" i stars and "design" j stars at the same time. Also, $n_i^c = \sum_j n_{ij}$ and $n_i^d = \sum_j n_{ji}$ (cost and design are shortened as "c" and "d" in this clear context). We focus our interest on these two aspects' rating correlation $Cor(r^c, r^d|n)$ before generalizing to any aspect pairs. First note

$$p_i^c = \sum_j p_{ij}, \qquad p_t^s = \sum_q p_{qt} \tag{3.6}$$

There are l sentiment levels $s_1, \ldots, s_l$, so

$$r^c = \sum_j s_i p_i^c = \sum_i \sum_j s_i p_{ij} = \sum_i \sum_j s_i p_{ij} \tag{3.7}$$

$$r^d = \sum_t s_t p_t^d = \sum_t \sum_q s_t p_{qt} = \sum_t \sum_q s_t p_{qt} \tag{3.8}$$

Following our Bayesian approach as in Section 3.3, we model probabilities $p_{11}, \ldots, p_{ll}$ by a Dirichlet posterior of parameters $(n_{11}+\alpha_{11}, \ldots, n_{ll}+\alpha_{ll})$. Denote $\gamma_{ij} = n_{ij}+\alpha_{ij}(i, j = 1, \ldots, l)$ and $\gamma_0 = \sum_{i,j} \gamma_{ij}$. We get variance $Var[p_{ij}]$ and co-variance $Cov(p_{ij}, p_{qt})$ in similar forms as Equation (3.2), (3.3).

$$Var[p_{ij}|n] = \frac{\gamma_{ij}(\gamma_0 - \gamma_{ij})}{\gamma_0^2(\gamma_0 + 1)} \tag{3.9}$$

$$Cov[p_{ij}, p_{qt}|n] = \frac{-\gamma_{ij}\gamma_{qt}}{\gamma_0^2(\gamma_0 + 1)} \qquad (ij \neq qt) \tag{3.10}$$

These are the building blocks to model $Cor(r^c, r^d|n)$.

$$
\begin{aligned}
Var(p_i^c|n) &= Var(\sum_j p_{ij}|n) \\
&= \sum_j Var(p_{ij}) + \sum_{j\neq t} Cov(p_{ij}, p_{it}) \\
Var(p_t^d|n) &= \sum_q Var(p_{qt}) + 2\sum_{j\neq q} Cov(p_{jt}, p_{qt}) \\
Cov(p_i^c, p_q^c|n) &= Cov(\sum_j p_{ij}, \sum_t p_{qt}|n) = \sum_{j,t} Cov(p_{ij}, p_{qt}) \\
Cov(p_j^d, p_t^d|n) &= Cov(\sum_i p_{ij}, \sum_q p_{qt}|n) = \sum_{i,q} Cov(p_{ij}, p_{qt}) \\
Cov(p_i^c, p_t^d|n) &= Cov(\sum_j p_{ij}, \sum_q p_{qt}|n) = \sum_{j,q} Cov(p_{ij}, p_{qt})
\end{aligned}
$$

Now we compute

$$
\begin{aligned}
Var(r^c|n) &= Var(\sum_i s_i p_i^c|n) \\
&= \sum_i s_i^2 Var(p_i^c) + \sum_{i\neq j} s_i s_j Cov(p_i^c, p_j^c) \\
Var(r^d|n) &= \sum_t s_t^2 Var(p_t^d) + \sum_{q\neq t} s_q s_t Cov(p_q^d, p_t^d) \\
Cov(r^c, r^d|n) &= Cov(\sum_j s_i p_i^c, \sum_t s_t p_t^d|n) \\
&= \sum_{i,t} s_i s_t Cov(p_i^c, p_t^d|n) = \sum_{i,t}\sum_{j,t} s_i s_t Cov(p_{ij}^c, p_{tk}^d)
\end{aligned}
$$

Finally, $Cor(r^c, r^d|n)$ can be estimated by Pearson correlation

$$
Cor(p^c, p^d|n) = \frac{Cov(r^c, r^d|n)}{\sqrt{Var(r^c|n) \times Var(r^d|n)}} \tag{3.11}
$$

This formula provides the correlation of any two aspects. We can then generalize an aspect's

uncertainty level provided in Equation (3.4) as

$$uncert_{a_i} = \min_{j=1,\ldots,m} \frac{Var(r^{a_j}|n)}{|Cor(r^{a_i}, r^{a_j}|n)|} \tag{3.12}$$

where a_i is an aspect. Note that, on the right hand side of above equation (3.12), when $j = i$, we have $Cor(r^{a_i}, r^{a_j}|n) = 1$. Hence, we get $Var(r^{a_i}|n)$ as a factor constituting $uncert_{a_i}$. The intuition behind Equation (3.12) is that we can take advantage of one aspect's rating to infer about the other's rating. Specifically, when two aspects are highly correlated, $|Cor(r^{a_i}, r^{a_j}|n)|$ is close to 1, thus the two aspects share the variance of the one with smaller variance.

We do not present the experimental results of this extended model as it does not show substantial improvement on key measurements so far. We doubt that it is due to the lack of a large dataset, though the model is in need of further study.

3.6 Experimental Evaluation

Our experiments were carried out on two real-world datasets: Amazon reviews provided by Bing Liu, et al. [31, 22], and Edmunds' car reviews that we crawled. We published our code, additional experiments and all used datasets on our supporting web page [43], for reproducibility purposes. The datasets were used to generate realistic sequences of reviews as described below.

The Amazon review dataset has been widely studied in the sentiment analysis community since it provides the ground-truth aspects and sentiments annotated manually by the authors. Moreover, different product types have different numbers of aspect. We omit products that have less than 4 aspects with at least 10 ratings, so we have enough aspects

Table 3.5: Dataset statistics.

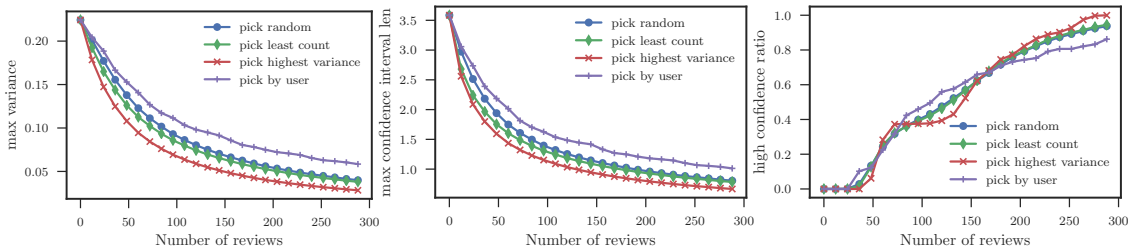
	Amazon reviews [31, 22]	Car reviews
#Products	8	501
#Sentiments (l)	6	5
#Reviews/product	51	106.67
#Aspects/product	4–21	7
#Ratings avg/aspect	27.31	77.76

for the algorithms to pick from and enough data to build a realistic rating distribution.

We crawled the second dataset using Edmunds' free open API on two car makes (Toyota and Honda) from 1990 to 2017, which resulted in 501 vehicles with 53,440 reviews in total. Our experiments were conducted on products that have at least 100 reviews (149 products). Furthermore, all vehicles share a fixed set of seven aspects: comfort, reliability, technology, value, performance, interior and safety. The datasets' characteristics are presented in Table 3.5.

In our evaluation, we start each experiment with no previous ratings information, and for each new simulated reviewer, we ask them to rate k aspects of a product. We conducted experiments with various k but only present the case of $k = 3$ due to space limitation; the results for other values of k followed similar trends.

Measures: Throughout all experiments, our first two measures are based on individual aspect rating's uncertainty level $uncert^{a_j}$. The first measure utilize the uncertainty value $Var[r^{a_j}|n]$ in Equation (3.4), which we explained why it is a reasonable measure in Section 3.3.2. To avoid biasing the results towards our selection algorithm that uses the aspects' variance, we introduced a second measure, which is the length of Confidence Interval (CI) of an aspect's ratings. The idea is that a smaller CI length means a higher degree of



(a) Max Variance (b) Max Confidence Interval Length (c) High Confidence Ratio (higher is better).

Figure 3.4: Comparing passive and active review solicitation (on Amazon reviews). Smaller is better, except for High Confidence Ratio measure.

confidence we know about an aspect's rating. In our experiment the CI is $\overline{X} \pm t(S/\sqrt{N})$, where $\overline{X}$ and S are the sample mean and variance of an aspect's ratings, respectively, N is the total number of ratings, t is the critical value specified by Student's t-distribution with $N-1$ degrees of freedom and confidence level $1-\alpha$. We choose confidence level 95% for all experiments.

Based on above measures, the key overall uncertainty measure we consider for a product is the maximum uncertainty among its aspects. Maximum is more appropriate than average, given our problem's motivation where we want to make sure that no aspect is left behind, that is, no aspect has too uncertain rating. Specifically, a product has multiple aspects $a_1, \ldots, a_m$, with uncertainty values $uncert^{a_1}, \ldots, uncert^{a_m}$, will have uncertainty level $\max_{j=1}^{m} uncert^{a_j}$.

As a third measure, we report the ratio of the number of aspects that we are confident about its rating statistics, thus we name this measure *"High Confidence Ratio"*. The idea is that when the confidence interval length of an aspect's ratings is smaller than

our desired threshold δ, then we can be confident about the aspect rating. We choose confidence level 95% and CI length threshold $\delta = 1$ for all experiments. High confidence ratio of 1 means that we are certain about all aspects' average rating. This measure reflects the degree of rating certainty instead of uncertainty as in the first two measures.

Since a dataset has multiple products, we report in the plots the uncertainty amount calculated by averaging uncertainty values over all products. In summary, we have three measures: "max variance", "max confidence interval length" and "high confidence ratio".

Baseline Aspect Selection Methods: Besides our proposed algorithm from Section 3.3.3, we consider two intuitive baseline methods used to pick k aspects to consult a new user: "*pick random*," which picks k random aspects from the set of an interested product's aspects, and "*pick least count*," which selects the k aspects with the least number of ratings so far. Given our toy example in Section 3.3.4, Table 3.2, *pick random* randomly selects four aspects with equal probability, whereas *pick least count* chooses aspect "design" first, then gives the three remaining aspects equal chances (because they have the same number of ratings: 33).

3.6.1 Active Versus Passive Solicitation

In the first experiment, we compare two approaches: letting the reviewer pick aspects to rate (passive, as in most existing Web sites) and actively asking them to rate specific aspects. In our datasets, the reviews of each product are fed to the various algorithm ordered by their generation timestamp. The result is presented in Figure 3.4, where a method asks a simulated user to rate k aspects. We refer to the user behavior in the traditional, passive solicitation as "*pick by user*" in the graphs. This method picks the first k aspects that

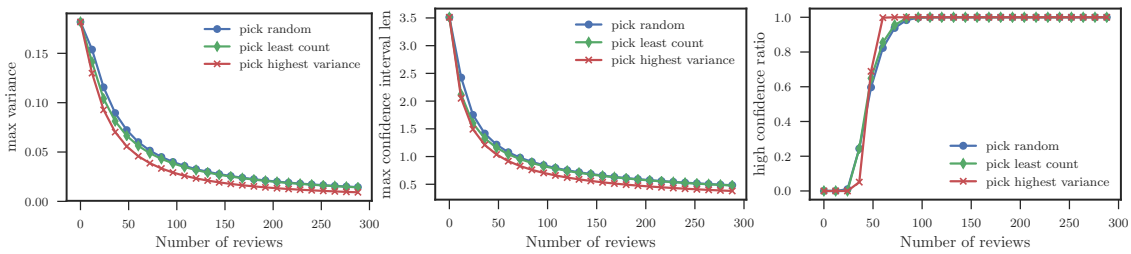
appear in the review under consideration. If a review has less than k aspects, we decrease the same number of solicited aspects for this position in all active methods for fairness.

We use the real reviews to realistically simulate the answers of the simulated user to the k selected aspects, as follows: we look up the sentiment of the asked aspect in the review currently under consideration if available. If the aspect is missing in the review, a simulated sentiment is computed from the rating distribution (which considers all reviews, not only the ones processed so far) of this aspect of this product. We refer to this rating scheme as "*answer almost real*" since it utilizes real user reviews in most cases.

We ran this experiment 200 times on all products independently, then take the average over all products. In each run, we solicit 300 reviews, up to $k = 3$ questions per review. If a product has less than 300 real reviews, we re-use its all available reviews to simulate answers. Since this experiment requires free-text review that is unavailable on our automobile dataset, we conducted it on Amazon review dataset only.

For all measures, we notice substantial improvements of the active methods over the passive solicitation method ("*pick by user*"). Illustrated by Figures 3.4(a), 3.4(b), and 3.4(c) respectively, the improvement is up to 52.6% for the "max variance", 34.7% for "max confidence interval length" and 14.4% for "high confidence ratio" measure with our "*pick highest variance*" method in the end of experiment. It is also worth noting that our method reaches the desired confidence on all aspects after about 270 reviews (when "high confidence ratio" is 1), while the passive method does not even reach this level by the end of the experiment (300 reviews).

The poor performance of "*pick by user*" is expected because users are normally



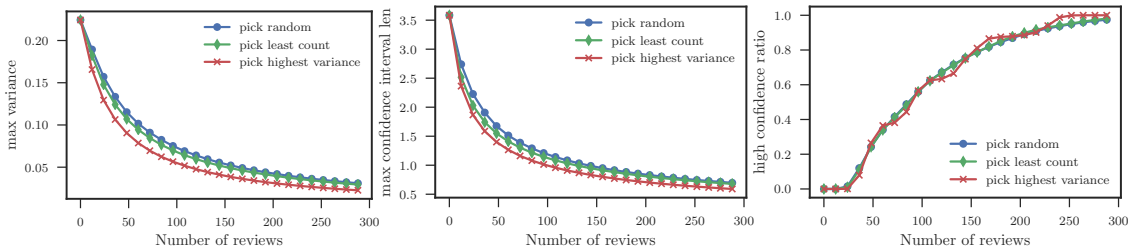
(a) Max Variance (b) Max Confidence Interval Length (c) High Confidence Ratio (higher is better)

Figure 3.5: Automobile reviews. Smaller is better, except for High Confidence Ratio measure.

biased toward common aspects with many ratings, while some aspects never get enough ratings to gain a reliable rating estimation. For example, for product "Nokia 6610," aspect "size" has around 210 ratings whereas "battery life" has only about 50 ratings, even though they have similar rating distribution shapes. Other methods distribute questions over aspects in a more balanced manner, thus get better performance. This result confirms our hypothesis that carefully selecting which aspects to ask users to rate can lead to higher review profile quality.

3.6.2 Comparison of Various "Active" Solicitation Methods

In this section, we compare our method "pick highest variance" to the two baselines, "*pick random*" and "*pick least count*," on both datasets. To scale to larger number of reviews and avoid the problem of the limited number of ratings for some aspects (e.g., "technology" and "safety" in the Edmunds dataset usually have less than 10 ratings per car), we consider a different answer generation scheme, where instead of using the real reviews one by one,



(a) Max Variance (b) Max Confidence Interval Length (c) High Confidence Ratio (higher is better)

Figure 3.6: Amazon reviews. Smaller is better, except for High Confidence Ratio measure.

we compute a ratings distribution for each aspect, and sample answers (ratings) from these distributions for each review. The experimental results are presented in Figure 3.5 and 3.6.

In both experiments, we solicit 300 reviews per product, 3 questions per review. We perform this simulation 200 times, then take the average for stable results. Our proposed method outperforms the two baselines consistently on both datasets and all measures. All methods start at the same point, then gradually diverge until the end of the experiments. By the end of the automobile reviews experiment, our method yields an uncertainty value that is 36.6% and 35.6% smaller than the value of "*pick random*" and "*pick least count*" accordingly in "max variance" measure (Figure 3.5(a) and 3.6(a)). The corresponding improvements in "max confidence interval length" measure are 21.5% and 20.9% (Figure 3.5(b) and 3.6(b)). In terms of confidence ratio, when our method reaches the full "high confidence ratio" (1) after about 60 reviews, the two baselines have the confidence ratio of 0.82 roughly and only reach full ratio after 90 reviews (Figure 3.5(c) and 3.6(c)).

The corresponding results in Amazon reviews present similar trends. Comparing to

the automobile review dataset under high confidence ratio measure, Amazon review dataset only has two differences. First, all methods reach the full ratio more slowly since Amazon products have larger aspect set, thus require more reviews. Second, our method's curve is smoother because Amazon products have a varying number of aspects instead of a fixed size (7 for Automobile products). Specifically, different number of product aspects result in different curves that are averaged to yield a pretty smooth curve as we observe.

It is worth mentioning that the two baseline methods behave slightly differently when the number of reviews performed is small; however, in the long run the number of times that aspects get selected evens out for both methods.

This is also a key difference between our method and baselines. Our method does not just ask about aspects equally as the baselines do. Instead, our method distributes more questions to aspects with contrasting ratings because these aspects need more information to solidify our belief of its rating. For instance, in toy example 3.2, the two baselines treat "weight," "cost" and "battery" equally (same rating counts), while our method "*pick highest variance*" would ask about "battery" first due to its polarized ratings.

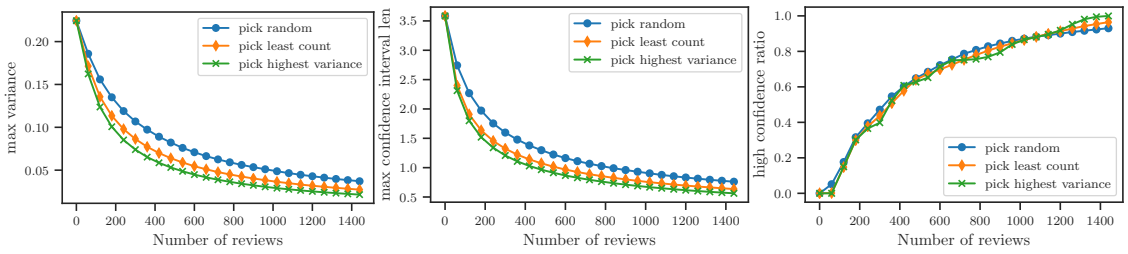
3.6.3 Extension to Response Probability

All previous experiments assume that users always provide their ratings. This assumption may not hold true in practice since users may not know about the solicited aspects, or just do not have time to respond to all. To reflect this fact, we present another set of experiments considering the probability that a user respond to the asked aspects. We estimate this response probability by counting all occurring reviews of an aspect, then normalized by the total number of all reviews in the dataset (Equation 3.5). Whenever an

aspect is selected to solicit users in our experiment, a simulated rating is returned only with above calculated response probability specific to that aspect.

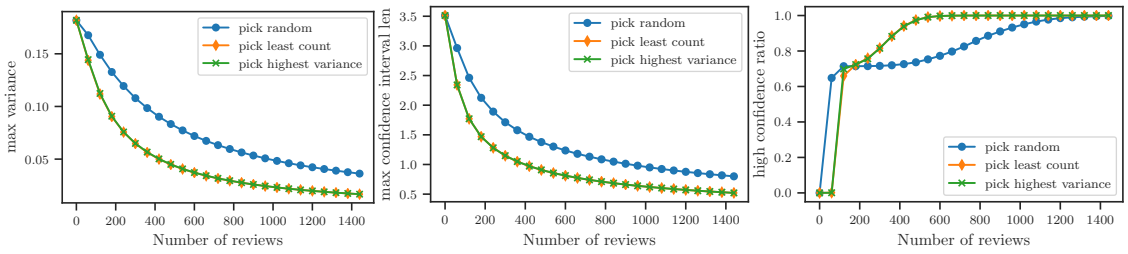
We present comparisons of various active solicitation methods, similarly to Section 3.6.2, but with this new response condition. Moreover, we solicit 1500 reviews per product to guarantee that all methods produce rating with high confidences on all aspects, i.e. "high confidence ratio" measure saturates at 1. Figure 3.7 shows the results for the Amazon review dataset. In the end of the simulation, our method "*pick highest variance*" achieves the lowest maximum variance, which is lower by 42.3% and 21.5% than "*pick least count*" and "*pick random*," respectively (Figure 3.7(a)). Moreover, "*pick highest variance*" reaches the maximal value for high confidence ratio measure after about 1420 reviews, while other methods can not achieve this even in the end of the simulation, or 1500 reviews (Figure 3.7(b)). It is also worth noting that after the same number of reviews, the uncertainty level in this experiment is much higher than the amount in previous experiment results in Figure 3.6. For example, after 300 reviews, the maximum variance of our method in this experiment is 0.074, while the similar value in Figure 3.6 experiment is about 0.025. The reason is that we only receive a user response with a probability in this experiment, thus a higher number of reviews is required to reach the same low uncertainty level as previously.

We present similar results on the automobile dataset in Figure 3.8. As usual, our "*pick highest variance*" beats the "*pick random*" baseline. It is interesting to notice that "*pick least count*" achieves similar performance comparing to our methods. This is because in this dataset there are two aspects "safety" and "technology, " which are rated only in 8% of all reviews, which is significantly lower than other aspects. Hence, both methods frequently



(a) Max Variance (b) Max Confidence Interval Length (c) High Confidence Ratio (higher is better).

Figure 3.7: Response with probability on Amazon review dataset. Smaller is better, except for High Confidence Ratio measure.



(a) Max Variance (b) Max Confidence Interval Length (c) High Confidence Ratio (higher is better).

Figure 3.8: Response with probability on automobile dataset. Smaller is better, except for High Confidence Ratio measure.

pick these aspects, as these two aspects have both the smallest number of ratings and the highest uncertainties.

3.6.4 Comparison of Hybrid Reviewing Interface to Passive Solicitation

In this experiment, we compare our proposed hybrid reviewing interface (Section 3.4) to the traditional passive solicitation method which employs free-form text re-

viewing only. Since this simulation requires text reviews which are not available in the automobile dataset, we only present results on the Amazon review dataset in Figure 3.9. In the plots, the passive solicitation is named "*pick free text only*" as it selects all aspects rated in the text review currently in consideration. We consider two variants of our hybrid reviewing interface: ask for one or three aspects (denote as "*pick highest variance 1/3 aspect*"). Similarly to Section 3.6.3, we also examine the case that users respond with a probability, which is denoted by suffix "response prob" in Figure 3.9. Note that this response probability is considered for additional aspect questions only. There is no need for generating rating for aspects mentioned in the current text review.

According to all measures, the hybrid reviewing interface significantly outperforms the passive solicitation method. The best performer is the hybrid interface with three additional aspects (pick highest variance 3 aspect), which has uncertainty level 70.9% and 47.7% lower than the baseline in max variance and max confidence interval length measures respectively. It also reaches the saturated high confidence ratio of 1 at an early stage (after 102 reviews), while the baseline needs more than 300 reviews. Unsurprisingly, the hybrid interface, when the response probability is 100%, outperforms the one when probability less than 100%. However, even in the presence of response probability, the hybrid interface with three additional aspects (pick highest variance 3 aspect response prob) achieves uncertainty that is 32.2% and 19.1% smaller than the baseline in max variance and max confidence interval length measures successively by the end of the experiment. The corresponding improvements of the hybrid interface with a single additional aspect are 25.1% and 14.7%.

Since the hybrid reviewing interface requires extra effort from users, we estimate

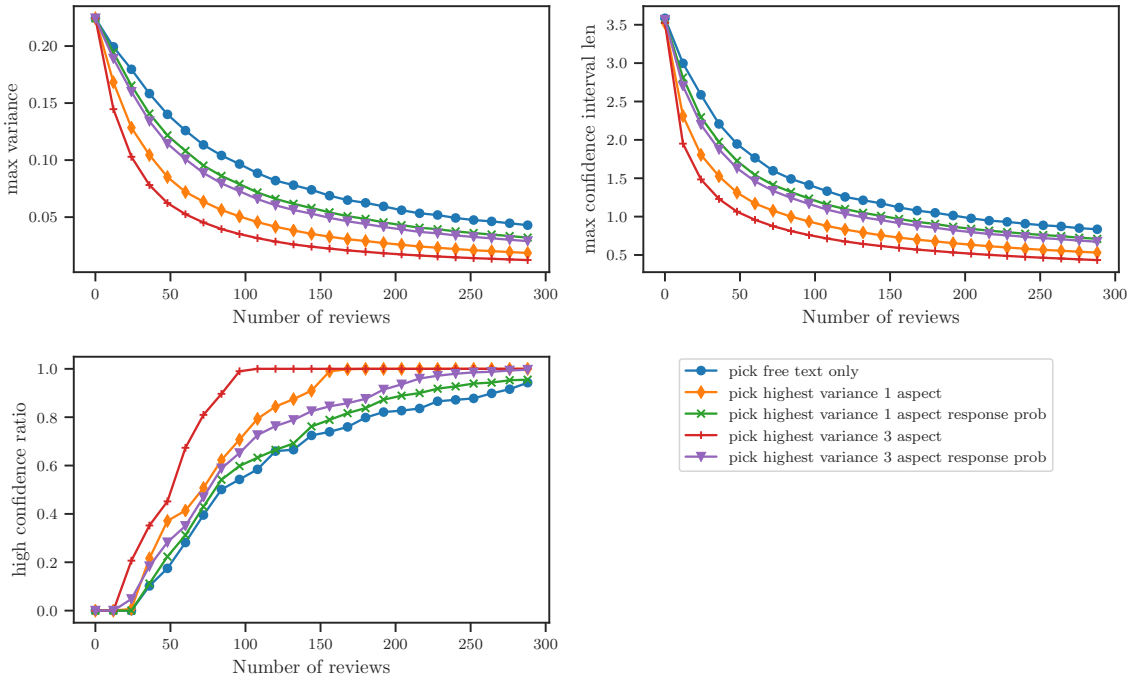


Figure 3.9: Hybrid reviewing interface on Amazon review dataset. Smaller is better, except for High Confidence Ratio measure.

that extra cost compared to the pure passive solicitation method. According to Furnham et. al. [25], in a study of 4943 participants, two online questionnaires including 206 and 154 items were completed on average in 1020.42 seconds and 915.69 seconds respectively, i.e. 4.95 seconds, 5.95 seconds per item respectively. Hence, we consider that users need on average 5.45 seconds to response per aspect. The typing speed was studied extensively in various settings. For example, in a survey by Arif and Stuerzlinger [3], Qwerty keyboard has an average 64.8 words per minutes (WPM). Kim et al. [38] reported a similar average number of 63 WPM for notebook and desktop keyboard setting. Recently, Ruan et al. [66] noted the typing speed of 53.46 WPM for mobile phone. We utilize the highest reported average speed (64.8 WPM) to continue our cost estimation. In the Amazon review dataset, there are 403.5 words per review on average, which require 6.2 minutes (or 372 seconds) on average per review. Note that, we only count sentences that mention at least one product aspect. Based on these numbers, we detail the time overhead against improvement of the hybrid reviewing interface comparing to passive solicitation in Table 3.6. We found that even with a single additional aspect question and considering response probability, the hybrid interface is able to decrease the max variance and max confidence interval length measure by 25.1% and 14.7% respectively with an extra cost (user time spent) of only 1.47%. For that reason, we argue that the hybrid reviewing interface is an efficient way to augment free-form text only interface to reduce rating uncertainty with little overhead.

Hybrid interface	Time overhead	Max variance	Max confidence interval len	High confidence ratio
1 additional aspect	1.47%	56.4%	36.1%	25.2%
3 additional aspects	4.41%	70.9%	47.7%	25.3%
1 additional aspect (w response prob)	1.47%	25.1%	14.7%	3.3%
3 additional aspects (w response prob)	4.41%	32.2%	19.1%	8.8%

Table 3.6: Time overhead and uncertainty reduction of hybrid review interface comparing to free-form text only interface after 300 reviews. For high confidence ratio measure, we cut off when the first method reaches saturation ratio of 1 (after 175 reviews)

3.7 Conclusions and Future Work

We have studied the problem of targeted review solicitation, which aims to achieve high-quality product review profiles, by actively soliciting aspects to rate. We adopted Bayesian inference statistics to model a review profile's key factors: product aspect rating estimation and its (un)certainty degree. We then introduced our algorithm to select k aspects to ask a new reviewer to optimize the review profile certainty. Using three different review profile quality measures, (variance, confidence interval length and high confidence ratio), we showed that our proposed active solicitation style clearly outperforms traditional passive solicitation methods on two real-world datasets. In another set of experiments our method beats two active solicitation baselines under all measures. Moreover, we propose a hybrid reviewing interface that incorporates active solicitation into passive approach with little extra user cost, while significantly reducing uncertainty. We further strengthen our experimental results with consideration of user response probability. To assist others reproducing our results, all our code and datasets are available online [43]. We also extended our model to account for correlated aspects.

In our future work, we plan to further estimate the user response probability using additional signals such as aspect correlation, user history and context. Another direction is to focus on rating uncertainty of discriminating aspects that best facilitating the comparison of competing products.

Chapter 4

Adaptive Goal-oriented Dialog Policy Generation using Dependency Graphs and Reinforcement Learning

4.1 Introduction

Dialog systems (a.k.a. chatbots) offer an intuitive and natural way for humans to interact with machines. Building intelligent dialog systems has been one the main goals of AI research since the inception of the field [35]. Early works on dialog systems use rule-based systems, while recent works are data-driven and use variants of deep learning models. Dialog systems can be categorized into two main categories: chit-chat and goal-oriented chatbots. Chit-chat bots are designed to engage in open-ended dialog with the goal of maintaining long and interesting conversations to the user, which has entertainment values. Example of

such a chit-chat bot is Microsoft's XiaoIce [88]. Goal-oriented chatbots, which are the focus of this work, facilitate achieving specific tasks, such as booking a flight ticket or ordering a pizza.

Despite the impressive advances in deep learning, commercial goal-oriented chatbots are mainly diagram-based, where the dialog between the chatbot and the user is modeled as a simple finite state machine. The key reason for the disconnect between the recent advances in deep learning based dialog systems such as seq2seq [73] inspired ones, and state-of-the-art commercial chatbot platforms like Google's DialogFlow [28] and Amazon Lex [2] is that seq2seq is suitable for chit-chat-style bots and may not be straightforwardly adapted to goal-oriented bots. State-of-the-art research goal-oriented chatbots use reinforcement learning (RL) to build their dialog policies [26], which may be flexible; however, they may violate the intended business logic, and they require large amounts of high quality training data to capture the business logic. While a chatbot diagrams is easy to design and execute, it only captures one possible conversation flow; there can be other valid conversation flows that are more suitable given a user behavior and a conversation context. To mitigate the rigidity of traditional chatbot diagrams, we propose *dependency graphs*, which are directed graphs that encapsulate all possible conversation flows. Our framework dynamically chooses the best valid conversation flow using reinforcement learning.

There can be, however, many other valid conversation flows A diagram encapsulates only one possible conversation flow While chatbots diagrams are easy to design and execute by a chatbot engine, they are rigid and they do not allow an execution engine to adapt to user characteristics.

Recent work on goal-oriented chatbots use reinforcement learning (RL), where the chatbot is modeled as an RL agent, and the user is modeled as the environment [26]. The actions of the RL model are the possible dialog acts the chatbot can make (e.g., greeting, inform, or request a slot value), and the states are the possible slot values collected from from the user in addition to the last dialog act generated. RL based chatbots are typically trained by bootstrapping using a limited number of conversations, and later improved by interacting with a user simulator. Existing RL based chatbots need to explore a large space of possible states before their dialog policy becomes mature.

However, asking the user to fill the right slots is not the only factors that determines the success of a chatbot. For example, the way that a question is asked or the answer is presented may affect the success of a chatbot, as discussed in `https://designguidelines.withgoogle.com/conversation/conversation-design/learn-about-conversation.html`. *We define success as walking users to commit their initial intents such as placing an order, booking a hotel room.*

We illustrate the difference of our success notion compared to traditional approach in an example in Figure 4.1. In the traditional chatbot, the focus is on filling accurately all slots such as pizza's type and size. Moreover, the implicit assumption is that the user will continue to the end to complete their order as long as the bot gets every slot correctly. However, there are many reasons that the user may drop out in the middle, even though their request is understood perfectly. For example, a promotion lover would like to enter or find out about coupon option and is disappointed about the lack of this option. Similarly, a conservative user would love to read some reviews, or see more pictures of the pizza before

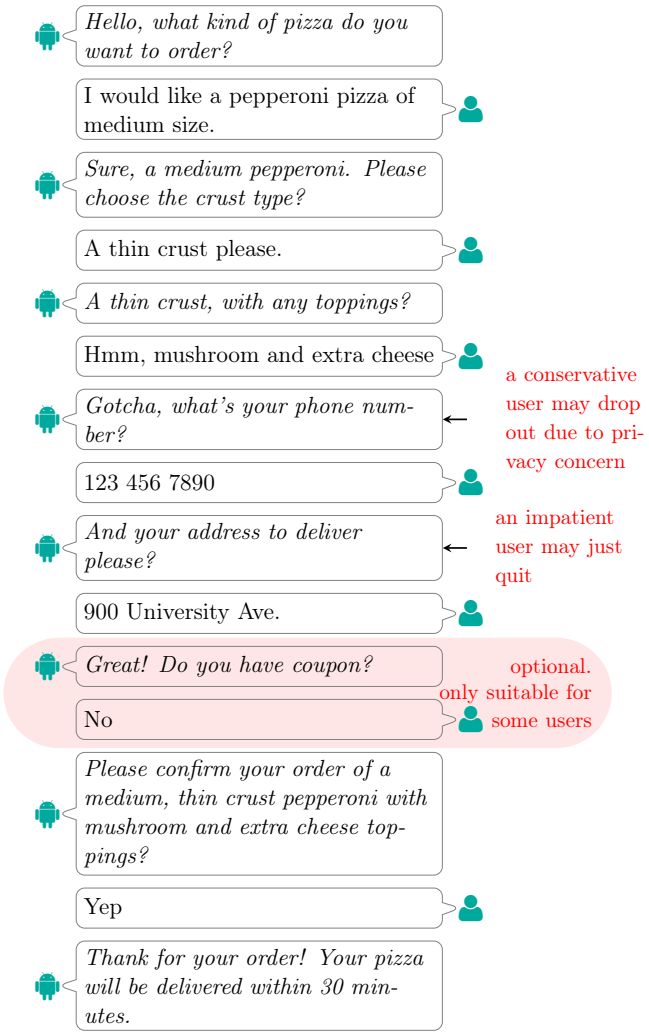


Figure 4.1: A pizza ordering chatbot conversation. Traditional chatbots focus on slot-filling (pizza type, size, and so on) and assume that the user continue to the end to finish their order. However, there is possibility that the user drops out in the middle due to various reasons.

placing the order. Of course, adding these features is nice but may backfire on other users due to the increase in conversation length, which is already 13. Moreover, a privacy-sensitive user may just drop out at utterance 9 since he/she is not willing to share phone number.

In these cases, even when the bot get the user's order accurately, the user may be still drop out instead of his/her initial buying intent. Therefore, to optimize for success rate,

the chatbot should also consider other affecting factors possibly specific to the user rather than merely slot-filling. As an analogy, our chatbot works like a sale man, helping customers successfully place pizza orders instead of just retrieving the customer's pizza specification.

In this chapter, we study the problem of goal-oriented chatbot optimization regarding to the aforementioned success definition. We borrow the well-studied factors of e-commerce web experience design [16] by mapping them into relevant optimization factors of a chatbot system (Table 4.1). For instance, uncertainty reducing elements in e-commerce website can be realized by showing reviews, information about product in the chatbot's dialogues. Further, the best way to conduct a dialog is user-specific. For example, providing additional information on the reviews of a pizza may be desirable for a user but not for another. Hence, we present a framework to define the high-level logic of chatbots and methods to automatically optimize its execution for each user or a group of similar users.

We first describe a goal-oriented chatbot system in an enriched model of the traditional slot-filing chatbot. Recall that, these traditional slot-filing models are guided by a diagram of nodes (slots), which specify the general order of slots to be solicited. Our enriched model groups semantically related nodes into a supernode, which should be solicited together. The model adds optional nodes targeting non-slot-filling factors such as a node providing reviews. Moreover, some nodes and supernodes may be constrained by the order of which nodes must go first. For example, the confirmation node must go last. This model of nodes and supernodes constrained by a dependency graph is the input for the chatbot optimizer. Similar to previous work, we consider this model as Markov Decision Process (MDP) [46] problem, or more advanced Partially Obserable MDP (POMDP) [65, 84] prob-

lem. To solve this, we employ reinforcement learning to find the optimal policy, i.e. find the best node to show the user given the current dialog history.

Our work differentiates from traditional approaches on three main points:

- We consider the possibility that a user drops out in the middle of conversation.
- We consider optional nodes regarding to specific users.
- We go beyond the limited setting of user profile in confirmation strategy selection by incorporating it into the chatbot's general policy optimization problem.

4.2 Dependency Graphs

Traditional Diagrams. Traditional goal-oriented chatbots attempt to acquire slot values from a user with the help of a diagram. The diagram guides a conversation by specifying what slots need to be filled and in what order; the diagram specifies the next slot to be filled based on the user's response when needed. Consider the example pizza ordering chatbot diagram in Figure 4.2. In this example, the diagram instructs the chatbot to ask for the slot values required to confirm a pizza order: pizza kind, size, crust type, and toppings. Also, the diagram specifies that the bot has to offer reviews to the user only if he responds with *yes* to the question: "Do you want to read reviews?"

Enriched Diagrams. We introduce *optional nodes* to enable selectively providing and soliciting information (slot values) that are not essential to the success of a conversation, but rather supplementary. Consider, for example, an impatient user trying to order a pizza.

Table 4.1: Chatbot's important factors mapped from Web domain [16]

Category	Web Site Factor	Chatbot Dependency Graph Node	Other Chatbot Functionality
Functionality factors - Usability	Convenience		need to login/register?
	Site navigation		Can go back, see progress
	Information architecture		
	Ordering/payment process		need to leave chatbot (go to web page)?
	Search facilities and process	NLU (e.g. big pizza = large pizza); confirmation policy (explicit, implicit)	
	Site speed		response time
	Findability/accessibility		is Messenger, SMS, Whatsapp?
Functionality factors - Interactivity	Customer service/after sales		follow-up after sale to check for problems
	Interaction with company personnel		agent can takeover chatbot
	Customization		
	Network effects		customer chat groups where they ask questions about product
Psychological factors - Trust	Transaction security	sentence explaining security	show encryption icon
	Customer data misuse	sentence explaining privacy policy	
	Customer data safety	sentence explaining privacy policy	
	Uncertainty reducing elements	show reviews/information about product	
	Guarantees/return policies	show refund policy; price match guarantee	
Content factors - Aesthetics	Design		if using existing medium, can play with multimedia (images/video) or medium-provided features like cards in Messenger; if Web-bot or mobile app, then also control the design of the chat window, e.g., show picture of agent who may assist or use nice colors
	Presentation quality		
	Design elements		
	Style/atmosphere		
Content factors - Marketing mix	Communication	show images of product	
	Product		
	Fulfillment		
	Price	say how much it is including tax and shipping (arrow to slots filling frame)	
	Promotion	show coupon info	
	Characteristics		

Offering such a user the chance to enter a coupon code might throw her off. In such a case, the chatbot engine should avoid asking the user about coupons and introducing optional nodes enables this functionality. We also introduce *supernodes* to group semantically related

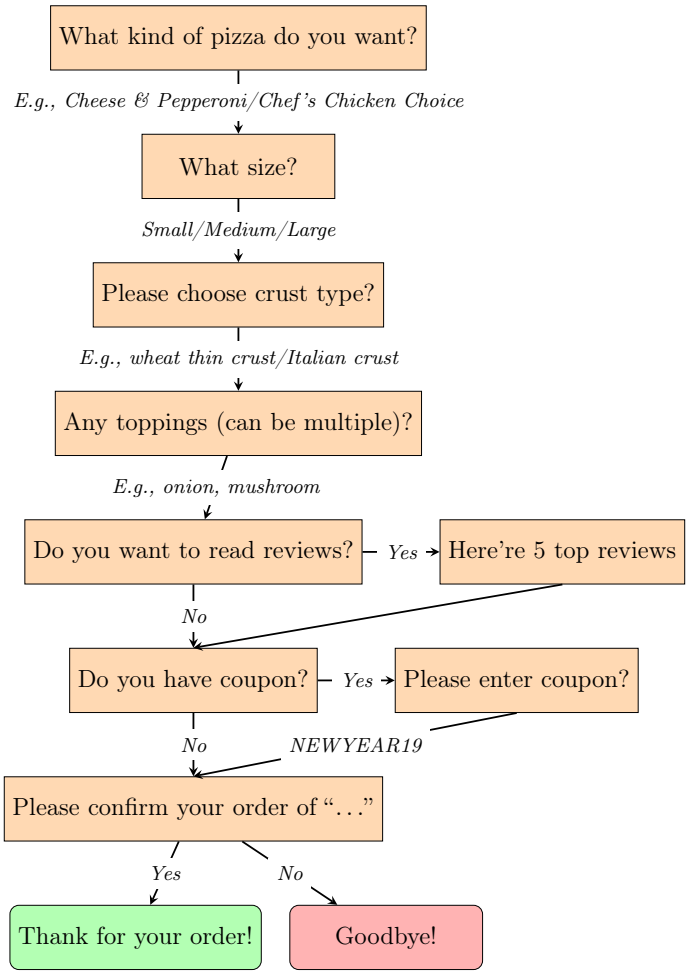


Figure 4.2: Example chatbot diagram: pizza ordering. Each node has a self-loop to allow repeatedly asking the same question until a valid answer is provided; we omit self-loops for brevity.

slots, similarly to frames in the frame-based paradigm [9]. Supernodes instruct a chatbot engine to solicit a group of slots either with one utterance or with a sequence of consecutive utterances. This is expected to result in a more natural dialogue. A supernode may impose restrictions on the order of filling its slots by means of a simple traditional diagram (within the supernode). A supernode can be optional, in which case it may contain *information blocks*. An information block can be a text message or an image and is used to facilitate

providing the user with extra information, if necessary, such as product reviews. Finally, we propose coupling supernodes with confirmation functionality. Confirmation is necessary to make sure the chatbot has captured the user's requirements appropriately. Confirmation in a supernode can be explicit or implicit. Figure 4.3 shows an enhanced diagram comprised of four supernodes, each with a group of related slots and a confirmation question.

Dependency Graphs. A specific diagram (or enriched diagram) is an instance of a conversation flow specification; there can be many equivalent diagrams. Consider the enhanced diagram in Figure 4.3: swapping supernodes v_2 and v_3 (i.e., asking the user if she has a coupon before asking if she wants to read reviews) conforms to the same specifications of the original enhanced diagram. However, the ordering of the nodes, thus the ordering of presenting questions/information to the user, is likely to affect user engagement. Designing engaging orderings without taking into account user characteristics and past experiences of the chatbot engine is challenging for the diagram designer. We propose replacing chatbot diagrams with *dependency graphs*, which are directed graphs that capture dependencies among supernodes rather than strict orderings. We show in Figure 4.4 an example dependency graph. This dependency graph does not impose restrictions on the order among supernodes $v_1 \dots v_3$, but rather imposes the restriction that the final confirmation (v_4) has to be presented only after the necessary information is collected. Dependency graphs offer two advantages over chatbot diagrams: They are easier to design, and they enable the chatbot engine the flexibility of choosing more engaging node orderings based on the user behavior.

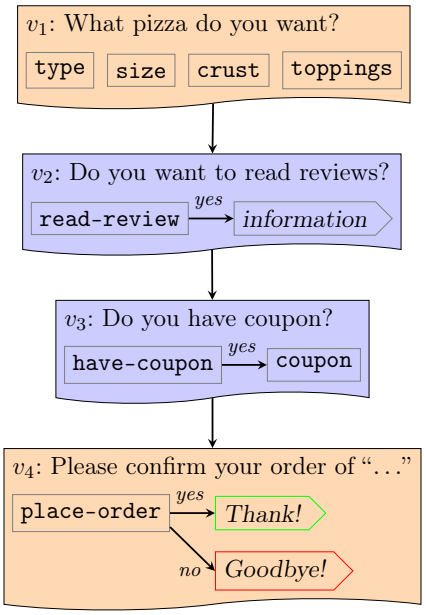


Figure 4.3: Enhanced chatbot diagram for pizza ordering. This diagram captures the requirements implied in the diagram in Figure 4.2, however, it is more concise and easier to design.

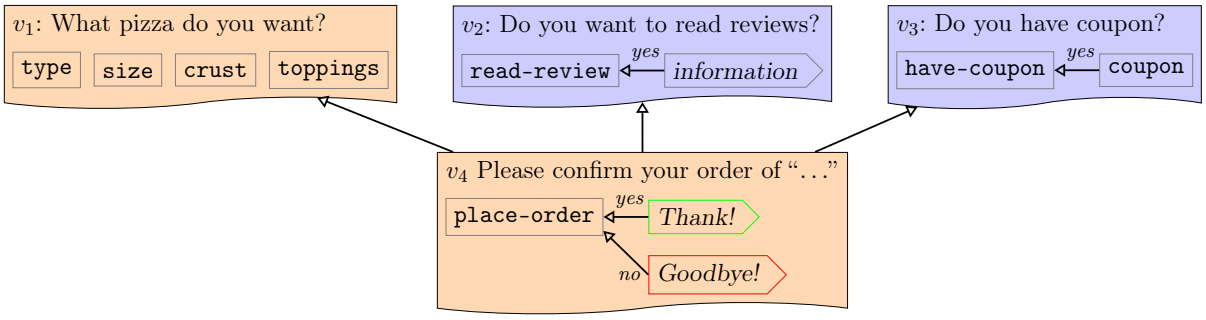


Figure 4.4: Dependency graph for pizza ordering. A directed edge from v_j to v_i means that v_j has a dependency on v_i; i.e., v_j should only be presented to the user after v_i.

4.3 Dialog Policy Generation

Figure 4.5 shows a high level overview of our policy generation framework. First, a domain expert (bot designer) creates a dependency graph that incorporates the application requirements and the intended dialog restrictions. Then, we will convert the dependency graph into an RL model, i.e., state space, action space, and reward function. Finally, we will

generate the dialog policy by incrementally training the RL model as follows. We bootstrap the model using a limited number of real chat logs (training data), which gives the RL agent a basic dialog policy. Then, we will use a simulator to generate a larger number of chat sessions, for various user personalities and contexts as described below. These sessions will continue training the RL model until a satisfactory dialog policy is generated.

In the *state space*, a state is a vector capturing the conversation history and any auxiliary information useful for the bot such as the number of matching items in the database. A typical state contains at least the following information: (a) the current value for each slot, e.g., pickuplocation="900 University Ave"; (b) a boolean variable for each slot specifying if it was filled by, or shown to the user; (c) three boolean variable for each supernode specifying if it is optional, activated or finished yet; (d) the user state and context, e.g., context="driving"; (e) the number of matching items in the database given the current slot constraints.

The *action space* consists of all available actions that the bot will select one per conversation turn to communicate with users. A basic action consists of a dialog act and a slot. The set of dialog acts include: request, inform, show, place-order, and possibly others if the task requires. Combining with the slots, here are a list of possible actions: (a) request user to fill a slot, e.g., "What insurance do you have?"; (b) inform user about a slot value, e.g., "there are three sizes of pizza: small, medium and large; (c) show an information block, e.g., "Your personal information is protected". Algorithm 5 shows how to convert a dependency graph into the state and action space.

The *reward function* assigns a score of 1 to the conversation success state, for

example, when the user completes making an appointment, and 0 otherwise. We will employ existing state-of-the-art RL policy generation algorithms [55] using this problem modeling.

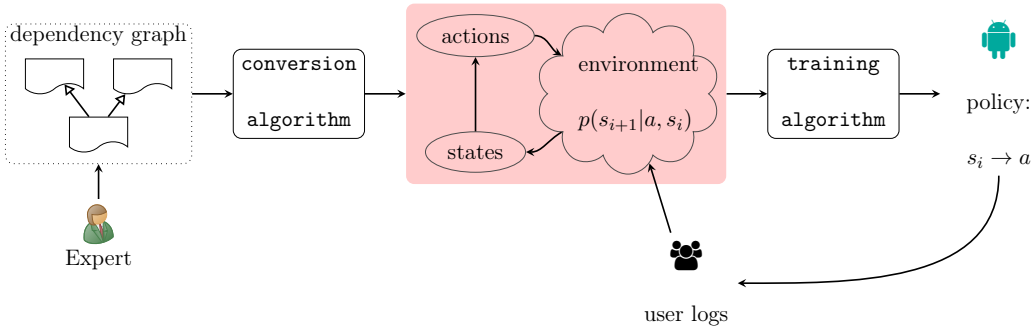


Figure 4.5: High level overview of our dialogue policy generation framework.

The dependency graph helps to reduce the action space by forbidding some actions at a specific state. First, slots in the same supernode have actions invoked together. In the other words, the bot cannot mix in the actions of slots of different supernodes. For example, when the bot is at the state s: (`fill_type` = True, `fill_size = True`, `fill_crust` = False, `fill_toppings` = False, `fill_read-review` = False, ...) (in supernode v_1 in Figure 4.4), the action `request_read-review` (supernode v_2) is illegal. Second, the dependencies also inhibit a set of actions. For instance, when the bot is again at the state s above, the bot has not finished supernode v_1, thus is not allowed to take action `request_place-order`, or `show_goodbye` of supernode v_4. Finally, inside each supernode, the action of a slot is only legal if the slot's dependencies have been satisfied. As an illustration, when the bot is in a state that has `fill_have-coupon` = False, the bot cannot take action `request_coupon`. We present our logic to check these dependency violation in Algorithm 6.

Algorithm 5 Convert dependency graph into state, action space.

Input: dependency graph $G(V, E)$

Output: states, actions

```
1: procedure MAP_STATE_SPACE(V)
2:     s = []                                                    ▷ state vector
3:     for v in V do
4:         for l in v.slots() do
5:             s.insert(fill_l, value_l)
6:         for b in v.infoblks() do
7:             s.insert(show_b)
8:         s.insert(is_optional_v,is_started_v, is_done_v)
9:     for i in user_contexts do
10:        s.insert(user_cont_i)
11:    Return s
12: procedure MAP_ACTION_SPACE(V)
13:    A = ∅                                                     ▷ action space
14:    for v in V do
15:        for s in v.slots() do
16:            A.insert(request_l, inform_l)
17:        for b in v.infoblks() do
18:            s.insert(show_b)
```

Algorithm 6 Check forbidden actions

Input: $G(V, E)$; state s, action a

Output: True or False

```
1: procedure VIOLATE_DEP_GRAPH(G(V, E), s, a)
2:     Return fail_slot_dependency(s, a) or
3:                 fail_supernode_dependency(s, a) or
4:                 cause_concurent_supernode(s, a)
```

```
1: procedure FAIL_SLOT_DEPENDENCY(s, a)
2:     l ← a.slot()
3:     for (prior_slot, val) in dependency(l) do    ▷ dependency(l) is a function returning slots and values that slot l depends on.
4:         if s.fill_prior_slot = False or s.value_prior_slot ≠ val then
5:             Return True
6:     Return False
```

```
1: procedure FAIL_SUPERNODE_DEPENDENCY(s, a)
2:     sn ← supernode_of(a)
3:     for prior_sn in sn_dependency(sn) do    ▷ dependency(sn) is a function returning supernodes that supernode sn depends on.
4:         if s.is_optional_prior_sn then
5:             if s.is_started_prior_sn & !s.is_done_prior_sn then
6:                 Return True
7:         else if !s.is_done_prior_sn then
8:             Return True
9:     Return False
```

```
1: procedure CAUSE_CONCURRENT_SUPERNODE(V, s, a)
2:     sn ← supernode_of(a)
3:     for other_sn in V do
4:         if s.is_started_prior_sn & !s.is_done_prior_sn then
5:             Return True
6:     Return False
```

4.4 User Simulation

Ideally, an RL agent learns from real-world feedback. In the context of goal-oriented chatbots, this means that the RL agent would have to do a possibly large number of conversations with real users before its dialogue policy becomes mature enough. Although crowd-sourcing services, such as Amazon MTurk, seemingly offer the possibility of training a chatbot by talking to MTurk workers, the time and financial cost of such a scheme can be prohibitive. Consequently, we rely on user simulation for training our RL agent, similarly to state-of-the-art research RL chatbot engines [69, 47, 48].

A user simulator should resemble real user behaviour: Users may get irritated by the chatbot utterances and choose to drop out, i.e., leave the conversation; they may provide irrelevant or uninterpretable utterances; or, ideally, users may provide the needed information, e.g., slot values. The probability of a certain user responding according to the mentioned scenarios is a function of her characteristics; estimating these probabilities is nontrivial due to their reliance on user characteristics. We denote the probability of user drop out, providing uninterpretable utterances, and providing relevant utterances by p_d, p_e, and p_s, respectively.

Our simulator determines the probability of each user response using a *user profile*, which is a set of parameters that describe a user. Inspired by research in online customer behavior [16] and the work in [69], we choose the following parameters to define a user profile: *security*, *privacy*, *uncertainty*, *guarantee*, *price*, and *promotion*. Each of these parameters is a numeric value between 0 and 1. *Security* and *privacy* quantify the level to which a user is concerned with her transaction security and data privacy, respectively. For

users with high values, the chance of dropping out if the chatbot does not explicitly provide security and privacy guarantees is high. *Uncertainty* quantifies a user's need for additional information, such as customer reviews, to clear his/her confusion. *Guarantee*, *price*, and *promotion* quantify a user's interest in product guarantees, price, and promotional coupons, respectively. More information about these aspects engages those users with high parameter values, but may cause others to get overwhelmed and drop out. A user u_j is described with a vector of her parameter values:

$$\mathbf{u}_j = \begin{bmatrix} security(u_j) \\ privacy(u_j) \\ uncertainty(u_j) \\ guarantee(u_j) \\ price(u_j) \\ promotion(u_j) \end{bmatrix}$$

.

We explain next the process of computing p_d, p_e, and p_s. These probabilities are functions of the interaction between a supernode in a dependency graph (which results in a chatbot utterance or a group of utterances) and a specific user profile. The user u_j is first initiated with a global user dropout rate: $p_d^0(u_j) = global\_dropout$. Then, the user dropout rate is adjusted based on her user profile as following.

$$p_d^0(u_j) = p_d^0(u_j) - \frac{1}{2} \times \sum \mathbf{u}_j$$

The subtraction penalizes the case that the bot fails to address user concern specified by

the user profile. Let

$$\mathbf{l}_i = \begin{bmatrix} security(l_i) \\ privacy(l_i) \\ uncertainty(l_i) \\ guarantee(l_i) \\ price(l_i) \\ promotion(l_i) \end{bmatrix}$$

be a vector of *relevance parameters* associated with a slot l_i. Each of these parameters is a numeric value between -1 and 1 that indicates the relevance of utterances generated by node l_i to a certain user parameter; negative values indicate an adversary effect, and positive values indicate an engaging effect. For example, if a user is highly security conscious (has a security parameter value of 1), and a supernode has a security relevance parameter of -1, the user is likely to drop out if the chatbot chooses to present her with utterances generated by this supernode. Let the *effect* of slot l_i to user u_j be the inner product of $\mathbf{l}_i$ and $\mathbf{u}_j$:

$$effect(l_i, u_j) = \mathbf{l}_i^\intercal \cdot \mathbf{u}_j$$

The user dropout rate after having received k chatbot utterances is updated to reflect the slot l_i's effect as below

$$p_d^k(u_j) = max(0, min(1, p_d^{k-1}(u_j) - \delta \times effect(l_i, u_j))) \tag{4.1}$$

where δ is a constant factor specifying the maximum dropout change per conversation turn. We further penalize the conversation logic violations that confuse users by a fix amount β:

$$p_d^k(u_j) = max(0, min(1, p_d^{k-1}(u_j) + \beta)) \tag{4.2}$$

An example of this violation is when the bot request users to place the order while have not collected all the pizza information. Note that, in above Equations, the value of $p_d^k(u_j)$ is always bounded between 0 and 1.

For simplicity, we assume that the probability of a user generating an uninterpretable utterance p_e is a constant ϵ bounded by the complement of the drop out probability:

$$p_e = min(\epsilon, 1 - p_d)$$

Finally, the probability of a user generating a valid utterance is: $p_s = 1 - p_d - p_e$.

4.5 Evaluation

Datasets: We evaluate our goal-oriented chatbot model using two tasks. The first task aims to help users booking movie tickets. For this, we utilize a real dataset collected via Amazon Mechanical Turk by Li et al. [47]. This dataset was created from 280 dialogues and has a database of 991 movies, out of which 133 user goals were randomly sampled for user simulation. The second task focuses on a pizza ordering system that we exemplified throughout the chapter. For this task, we build a database of pizzas by collecting pizza specifications from Dominos; for example, a "small" pizza has two crust options: "hand tossed" and "gluten free crust". We sample 10,000 user goals from this pizza database. Users can inform a slot's value to the bot or request the bot for available options of a slot. Table 4.2 reports more details of these two datasets.

Methods: We design our method's dependency graphs very similar to the one in Figure 4.3. Besides having more complex pizza supernode (v_1) and only two optional supernodes on review reading, coupon, we have other optional supernodes addressing user concerns on privacy, security, guarantee and final price details. After converting into an RL model, we employ Deep Q-Network [55] to train the agent using user simulated conversations.

	Movie ticket booking	Pizza ordering
Slot	moviename, theater, starttime, date, genre, state, city, zip, critic-rating, mpaa-rating, distanceconstraints, video-format, theater-chain, price, actor, description, other, numberofkids, numberofpeople	size, crust, cheese, sauce, sauce level, toppings
Number of items	991	16,560
Number of user goals	133	10,000

Table 4.2: Dataset details.

While we are using DQN as the base, any RL algorithms can be utilized.

We compare our method, named *RL with DepGraph*, with several baselines. The first baseline, *RL with DepGraph*, is a straight-forward RL model. Since this model is not imposed by a dependency graph, it only learns the dependency violations via user negative reaction, which is an increase of user dropout rate in our user simulation. Another set of baselines is based on hand-crafted rules by the bot designers, also known as diagram-based models. In particular, there are four variants listed below depending on the way of choosing the next slot to request users and whether to include optional slots.

- *Sequence Rule*: select the next slot in a predefined sequence, include all optional slots.
- *Random Rule*: randomly select the next slot from the un-filled ones and include all optional slots.
- *Sequence/Random Rule* - No Optional: same as the Sequence, Random Rule respectively but do not include optional slots.

These rule-based methods also support user requests by simply informing available values of the requested slots.

Measures: Similar to Li et al. [47, 48], we consider conversation's success rate as the single most important measure. As we explained in the Introduction, this measure reflects our chatbot's design goal of adaptively and precisely walking users to reach their initial intents, i.e. booking the movie ticket or ordering pizza. Note that violating the slots dependency, such as showing the placing-order confirmation before finishing all pizza information, is considered as a failure case since it confuses the users. Our experiment is separated into the training phase, which applies to only RL based methods, and the testing phase that applies for all methods. For the training phase, we report the success rate changes over the number of simulated conversations. For the testing phase, we show the performances of all methods after 2500 simulated conversations.

Experiment's Parameters: We carry out our experiments on a set of different user profiles, which is explained in Section 4.4. Specifically, there are six user factors, each is set to 0 or 1, therefore there are $2^6 = 64$ different user profiles in total. We run our experiments on all user profiles and report the average results. Since we have a fix number of user goals to initiate simulated users, we randomly partition these goals into train/test set of ratio 9/1. In this manner, we make sure that we do not have simulated users in both training and testing phase with the same user goal. We list other experiment details in Table 4.3.

4.5.1 Performance in the Training Phase

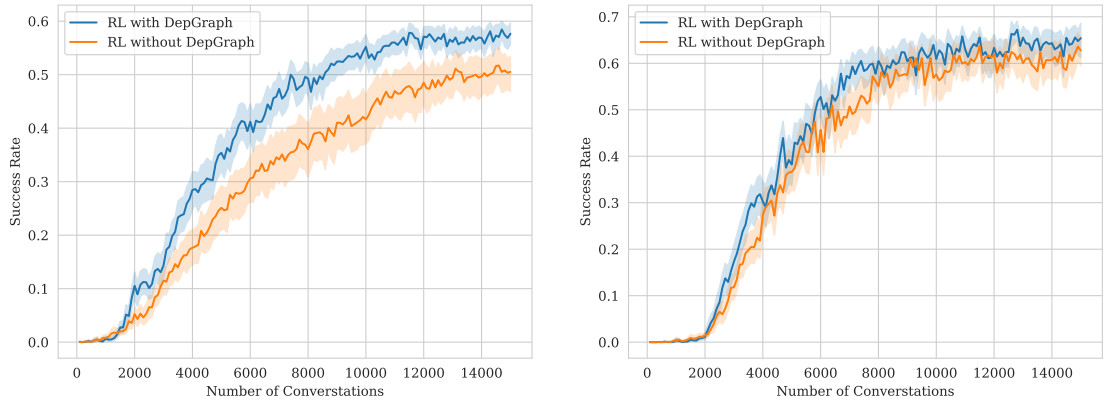
We train our method model and the RL baseline using 15,000 simulated conversations. Note that the other rule-based methods do not require any training, thus are not

Parameter	Value	Description
Global initial dropout	3%	the base dropout rate, 3% means that only $1 - 0.97^{20} \approx 0.456$, or 45.6% chance that a user has not dropped out after 20 turns.
Dropout delta (δ)	1%	the dropout multiplier in user change in response to the effect of the presented slots (Equation 4.1).
Dropout increase due to violation (β)	3%	user's dropout is increase as she is confused by the bot due to dependency violation (Equation 4.2).
Slot error rate (p_e)	5%	Artificial error rate introduced to mimic the user misunderstanding and NLU's mistakes [47, 48].

Table 4.3: Experiment's Parameters.

presented in this section. We present our results for both datasets in Figure 4.6. The average success rates are shown in the solid line, while their variations are the shaded regions. For both datasets, our method achieves a higher success rate and saturates much faster than the RL baseline. In particular, our method outperforms the RL baseline by 20.5% and 6.7% on average on movie and pizza dataset respectively. The improvement in the movie dataset is more substantial than that of the pizza dataset since the movie ticket booking task is more complex, i.e. longer conversations and bigger set of slots, thus has more rooms for improvement. For the same reason, the success rate in the movie dataset is higher than in the pizza dataset.

We further examine how two methods select available optional slots to understand their performances. Due to the space limitation, We only show statistics for the movie dataset in Figure 4.7. A method is well adaptive to a user when it can pick up appropriate optional slots to address the user concern. For instance, if a user is interested in promotion, then showing the user coupon options would motivate the user so that reducing her dropout



(a) Movie dataset

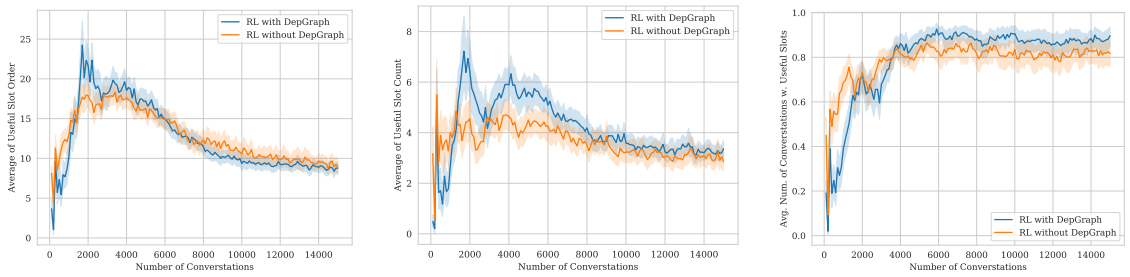
(b) Pizza dataset

Figure 4.6: Success rate changes over the number of simulated conversations in the training phase.

rate. In Figure 4.7(a), we show that our method learns to select useful slots earlier in the conversations than the RL method. This is helpful to benefit from a lower user dropout rate as soon as possible. In Figure 4.7(b) and 4.7(c), we report the average number of useful slots per conversation, and the average ratio of conversations that the bot successfully pick at least a useful slot. For both statistics, the higher is better since it means that the user is presented with appropriate slots. Our method has higher numbers in both cases, thus achieves a higher success rate as being shown in Figure 4.6.

4.5.2 Performance in the Testing Phase

In the testing phase, we test all methods using 2,000 conversations simulated from the testing set of user goals. We present the success rates for both datasets in Figure 4.8. In both cases, our method outperforms all methods, especially the rule-based ones. Comparing to the RL agent, our method's success rate is 12.7% and 6.2% higher on average on movie



(a) Average order of useful slots in the conversation.

(b) Average number of useful slots per conversation.

(c) Avg. number of conversations containing a useful slot.

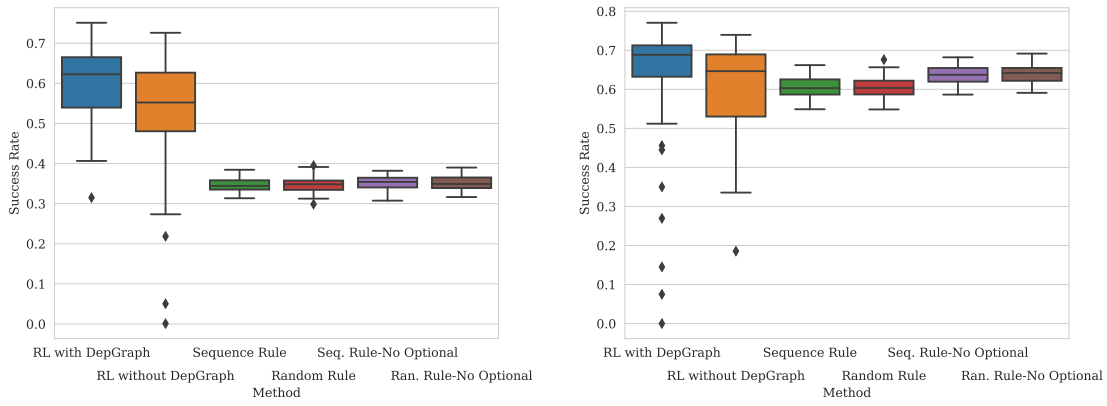
Figure 4.7: Method's behaviors in picking up optional slots.

and pizza dataset respectively. All rule-based methods' performances are quite similar and lower than both our and the RL method. In particular, our method achieves an improvement of about 70% and 11% on average over the best rule-based method on the movie and pizza dataset subsequently. Since the pizza ordering task is shorter and simpler than the movie booking one, all methods perform reasonably. The results also indicate that our method can scale to more complicated conversation scenarios such as the movie ticket booking one.

4.6 Related Work

Differ from social conversational dialog system, users in goal-oriented chatbot have a specific goal to complete. While goal-oriented chatbot has been studied for decades, the recent advances in Deep Learning, Natural Language Processing have brought new tools, increased interests.

Chatbot Policy Management: Traditional goal-oriented dialogue systems rely



(a) Movie dataset

(b) Pizza dataset

Figure 4.8: Success rate changes over the number of simulated conversations in the testing phase.

on slot filling and dialog act to model conversations. Bobrow et al. [9] proposed an influential model that organizes related slots into frames. The classical models control the conversation flow by a diagram or set of hand-crafted rules carefully designed by a domain expert. Another main approach models [46, 70] the goal-oriented chatbot as a Markov Decision Process (MDP) and utilize Reinforcement Learning model to learn a conversation policy. Following these early systems, more recent efforts [64, 84, 87, 75] utilize more complicated Partially Observed MDP (POMDP) model. More advanced RL algorithms customized for conversational are also proposed [13]. In overall, all above systems share a common modular framework that has four modules: Natural Language Understanding (NLU), Dialog State Tracker (DST), Dialog Policy (DP) and Natural Language Generation (NLG). Closely related to the DP, DST is a challenging NLP topic attracts a lot of community attention [30, 83].

Different from this modular framework, recently proposed end-to-end chatbots [81, 11, 19] take a user utterance as input and output a bot response. The challenge of these

systems is to model the database and knowledge graph that are normally parts of goal-oriented dialogue systems. While this is a promising approach, there is still a lack of complete comparison between it and the traditional approach.

User Simulation: There are several popular approaches simulating user behaviors to train the chatbot policy. Eckert et al. [23] propose a simple N-gram simulation that basically sample a user action based on previous N-1 actions of the bot and user. Agenda-based user simulation [68, 36] relies on user goal to keep consistent context, and an agenda which is a stack of user actions to realize. A recent approach [60] model user actions based on Bayesian network.

Dialog System Platform Amazon Lex [2], Google Dialogflow [28] are popular industrial chatbot frameworks using advanced neural based NLU, NLG modules, and classical rule-based dialog managers. In acamedia, TC-Bot [47, 48] is the first open source end-to-end implementation that supports reusable NLU, NLG and dialog management modules. ConvLab [45] can be considered as a TC-Bot's successor supporting multi-domain environments. Our implementation is based a simplified version of TC-Bot focusing on dialog policy generation only[1].

4.7 Conclusions

In this chapter, we proposed a novel, hybrid dialog manager for goal-oriented conversational agents that is is interpretable to the bot designer, supporting rich semantics and flexible dialogue scenarios. Our key idea is the dependency graph, by which a bot designer

[1]https://github.com/maxbren/GO-Bot-DRL

can capture the conversation constraints intuitively. Our algorithm transforms this dependency graph into an RL model that is trained on user chat logs to materialize an adaptive dialog policy. We evaluated the proposed method with the standard Agenda-based user simulation augmented with the notion of user profiles and user dropout for more realistic user modeling. Our method's results on the movie ticket booking and pizza ordering datasets showed are promising. In the training phase, our method saturated faster and achieved higher success rate that the RL based method. In the testing phase, our method outperforms all baseline methods based on diagram and RL. We showed that our model adapts to users, thus chooses appropriate slots to walk users to complete their goals.

Chapter 5

Conclusions

In this dissertation, we have exploited the semantics embedded in user-generated text to enable efficient analysis and retrieval. Relying on advances in sentiment analysis and information extraction to extract semantics, we showed how to take advantage of this information to better organize, retrieve user-generated data, and utilize it to incrementally improve the systems. In this context, this dissertation's contributions are three-fold.

First, we presented a fresh review summarization framework that advances the state-of-the-art by proposing a coverage definition that encapsulates both concept meaning and sentiment. In particular, our coverage leverages a domain hierarchy of concepts to handle the semantic overlap among the aspects and further constrains aspects of similar sentiment levels. We proved that the problem is NP-hard and presented bounded approximate algorithms to compute the most representative set of sentences or reviews. In our quantitative evaluation, we showed that the Greedy algorithm achieves review coverage comparable to the optimal algorithm but in a much shorter time. We further made use of various in-

tuitive summary quality measures and demonstrated that the Greedy outperforms several popular baselines on selecting k sentences to summarize real reviews.

Second, we studied the problem of dynamically selecting a focused set of aspects to obtain users' annotations for high-quality product review profiles. We presented a principled approach to account for the profile's factors of rating estimation and uncertainty based on Bayesian statistics. Using this framework, we proposed an algorithm to choose k aspects to ask a reviewer given the current product rating history. We showed that this method outperforms both the traditional passive solicitation style and other active solicitation baselines under various review profile quality measures on two real-world datasets. Furthermore, we extended our selection algorithm to work under different settings and to consider the probabilistic factor in user response. Particularly, we proposed a hybrid reviewing interface that augments the traditional approach with active solicitation at little extra user cost and at the same time, with significantly lower rating uncertainty.

Finally, we studied how to utilize conversational chat logs to learn adaptive workflows of goal-oriented chatbots. We proposed a hybrid dialog manager model that bridges two traditional approaches (frame-based and Reinforcement Learning based) to take advantages of their interpretability, rich semantic support and flexible conversation scenarios. Our chatbot framework's only input, the Dependency Graph, is easy to design and intuitive to validate. From there, our algorithm can learn personalized chatbot workflows to maximize their success rate. We based our evaluation on a real dataset of a movie ticket booking system using the popular Agenda-based user simulation. In the training phase, we found that our method achieves a higher success rate at a faster speed than the baseline RL

based method. In the testing phase, we also noted clear improvements of our model over all baseline methods.

In summary, we have shown that leveraging the embedded semantic is a powerful tool for mining user-generated text. With the advancement of NLP techniques to extract high-quality text semantics, there are great opportunities for higher-level studies and applications exploiting this information.

Bibliography

[1] Alan Agresti and David B Hitchcock. Bayesian inference for categorical data analysis. *Statistical Methods & Applications*, 14(3):297–330, 2005.

[2] Amazon. Amazon lex. `https://aws.amazon.com/lex/`, 2019.

[3] Ahmed Sabbir Arif and Wolfgang Stuerzlinger. Analysis of text entry performance metrics. In *2009 IEEE Toronto International Conference Science and Technology for Humanity (TIC-STH)*, pages 100–105. IEEE, 2009.

[4] Alan R Aronson. Effective mapping of biomedical text to the umls metathesaurus: the metamap program. In *Proceedings of the AMIA Symposium*, page 17. American Medical Informatics Association, 2001.

[5] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[6] Hyunmi Baek, JoongHo Ahn, and Youngseok Choi. Helpfulness of online consumer reviews: Readers' objectives and review cues. *International Journal of Electronic Commerce*, 17(2):99–126, 2012.

[7] Michal Belica. Sumy: Module for automatic summarization of text documents. `https://pypi.python.org/pypi/sumy`, 2017.

[8] Sasha Blair-Goldensohn, Kerry Hannan, Ryan McDonald, Tyler Neylon, George A Reis, and Jeff Reynar. Building a sentiment summarizer for local service reviews. In *WWW workshop on NLP in the information explosion era*, volume 14, pages 339–348, 2008.

[9] Daniel G Bobrow, Ronald M Kaplan, Martin Kay, Donald A Norman, Henry Thompson, and Terry Winograd. Gus, a frame-driven dialog system. *Artificial intelligence*, 8(2):155–173, 1977.

[10] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl 1):D267–D270, 2004.

[11] Antoine Bordes, Y-Lan Boureau, and Jason Weston. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*, 2016.

[12] Giuseppe Carenini, Jackie Chi Kit Cheung, and Adam Pauls. Multi-document summarization of evaluative text. *Computational Intelligence*, 29(4):545–576, 2013.

[13] Inigo Casanueva, Paweł Budzianowski, Pei-Hao Su, Stefan Ultes, Lina Rojas-Barahona, Bo-Hsiang Tseng, and Milica Gašić. Feudal reinforcement learning for dialogue management in large domains. *arXiv preprint arXiv:1803.03232*, 2018.

[14] Kuang Chen, Harr Chen, Neil Conway, Joseph M Hellerstein, and Tapan S Parikh. Usher: Improving data quality with dynamic forms. *IEEE Transactions on Knowledge and Data Engineering*, 23(8):1138–1153, 2011.

[15] Marek Chrobak, Claire Kenyon, John Noga, and Neal E Young. Oblivious medians via online bidding. In *LATIN 2006: Theoretical Informatics*, pages 311–322. Springer, 2006.

[16] Efthymios Constantinides. Influencing the online consumer's behavior: the web experience. *Internet research*, 14(2):111–126, 2004.

[17] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3079–3087, 2015.

[18] Dipanjan Das and André FT Martins. A survey on automatic text summarization. *Literature Survey for the Language and Statistics II course at CMU*, 4:192–195, 2007.

[19] Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. Towards end-to-end reinforcement learning of dialogue agents for information access. *arXiv preprint arXiv:1609.00777*, 2016.

[20] Adji B Dieng, Chong Wang, Jianfeng Gao, and John Paisley. Topicrnn: A recurrent neural network with long-range semantic dependency. *arXiv preprint arXiv:1611.01702*, 2016.

[21] Angelika Dimoka, Yili Hong, and Paul A Pavlou. On product uncertainty in online markets: Theory and evidence. *MIS quarterly*, 36:395–426, June 2012.

[22] Xiaowen Ding, Bing Liu, and Philip S Yu. A holistic lexicon-based approach to opinion mining. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 231–240. ACM, 2008.

[23] Wieland Eckert, Esther Levin, and Roberto Pieraccini. User modeling for spoken dialogue system evaluation. In *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, pages 80–87. IEEE, 1997.

[24] Günes Erkan and Dragomir R Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479, 2004.

[25] Adrian Furnham, Gillian Hyde, and Geoff Trickey. On-line questionnaire completion time and personality test scores. *Personality and Individual Differences*, 54(6):716–720, 2013.

[26] Jianfeng Gao, Michel Galley, Lihong Li, et al. Neural approaches to conversational ai. *Foundations and Trends® in Information Retrieval*, 13(2-3):127–298, 2019.

[27] Jade Goldstein, Vibhu Mittal, Jaime Carbonell, and Mark Kantrowitz. Multi-document summarization by sentence extraction. In *Proceedings of the 2000 NAACL-ANLPWorkshop on Automatic summarization-Volume 4*, pages 40–48. Association for Computational Linguistics, 2000.

[28] Google. Google dialogflow. `https://dialogflow.com/`, 2019.

[29] Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2015.

[30] Matthew Henderson, Blaise Thomson, and Jason D Williams. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272, 2014.

[31] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

[32] SNOMED International. SNOMED CT. `https://www.snomed.org/snomed-ct`, 2016.

[33] Niklas Jakob and Iryna Gurevych. Extracting opinion targets in a single-and cross-domain setting with conditional random fields. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 1035–1045. Association for Computational Linguistics, 2010.

[34] Nitin Jindal and Bing Liu. Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 219–230. ACM, 2008.

[35] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*, chapter Dialog Systems and Chatbots. preprint, 2017.

[36] Simon Keizer, Milica Gašić, Filip Jurčíček, François Mairesse, Blaise Thomson, Kai Yu, and Steve Young. Parameter estimation for agenda-based user simulation. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 116–123. Association for Computational Linguistics, 2010.

[37] Adwait Khare, Lauren I Labrecque, and Anthony K Asare. The assimilative and contrastive effects of word-of-mouth volume: An experimental examination of online consumer ratings. *Journal of Retailing*, 87(1):111–126, 2011.

[38] Jeong Ho Kim, Lovenoor Aulck, Michael C Bartha, Christy A Harper, and Peter W Johnson. Differences in typing forces, muscle activity, comfort, and typing performance among virtual, notebook, and desktop keyboards. *Applied ergonomics*, 45(6):1406–1413, 2014.

[39] Suin Kim, Jianwen Zhang, Zheng Chen, Alice H Oh, and Shixia Liu. A hierarchical aspect-sentiment model for online reviews. In *AAAI*, 2013.

[40] Youngsoo Kim and Ramayya Krishnan. On product-level uncertainty and online purchase behavior: An empirical analysis. *Management Science*, 61(10):2449–2467, 2015.

[41] Nikolaos Korfiatis, Elena GarcíA-Bariocanal, and Salvador SáNchez-Alonso. Evaluating content quality and helpfulness of online product reviews: The interplay of review helpfulness vs. review content. *Electronic Commerce Research and Applications*, 11(3):205–217, 2012.

[42] Theodoros Lappas, Mark Crovella, and Evimaria Terzi. Selecting a characteristic set of reviews. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 832–840. ACM, 2012.

[43] Nhat Le, Ryan Rivas, James Flegal, and Vagelis Hristidis. Supporting webpage. `www.cs.ucr.edu/~nle020/review_solicitation/`, 2018.

[44] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196, 2014.

[45] Sungjin Lee, Qi Zhu, Ryuichi Takanobu, Xiang Li, Yaoqin Zhang, Zheng Zhang, Jinchao Li, Baolin Peng, Xiujun Li, Minlie Huang, and Jianfeng Gao. Convlab: Multi-domain end-to-end dialog system platform. *ArXiv*, abs/1904.08637, 2019.

[46] Esther Levin, Roberto Pieraccini, and Wieland Eckert. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on speech and audio processing*, 8(1):11–23, 2000.

[47] Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. End-to-end task-completion neural dialogue systems. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 733–743, 2017.

[48] Xiujun Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688*, 2016.

[49] Bing Liu and Lei Zhang. A survey of opinion mining and sentiment analysis. In *Mining text data*, pages 415–463. Springer, 2012.

[50] Yue Lu, ChengXiang Zhai, and Neel Sundaresan. Rated aspect summarization of short comments. In *Proceedings of the 18th international conference on World wide web*, pages 131–140. ACM, 2009.

[51] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics, 2011.

[52] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004.

[53] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[54] Takeru Miyato, Andrew M Dai, and Ian Goodfellow. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*, 2016.

[55] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[56] Susan M Mudambi and David Schuff. Research note: What makes a helpful online review? a study of customer reviews on amazon. com. *MIS quarterly*, 34:185–200, 2010.

[57] Arjun Mukherjee and Bing Liu. Aspect extraction through semi-supervised modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 339–348. Association for Computational Linguistics, 2012.

[58] Roberto Navigli and Simone Paolo Ponzetto. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250, 2012.

[59] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

[60] Olivier Pietquin and Thierry Dutoit. A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(2):589–599, 2006.

[61] Ana-Maria Popescu and Orena Etzioni. Extracting product features and opinions from reviews. In *Natural language processing and text mining*, pages 9–28. Springer, 2007.

[62] Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. Opinion word expansion and target extraction through double propagation. *Computational linguistics*, 37(1):9–27, 2011.

[63] Dragomir R Radev, Hongyan Jing, and Malgorzata Budzikowska. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *Proceedings of the 2000 NAACL-ANLP Workshop on Automatic summarization*, pages 21–30. Association for Computational Linguistics, 2000.

[64] Verena Rieser and Oliver Lemon. *Reinforcement learning for adaptive dialogue systems: a data-driven methodology for dialogue management and natural language generation.* Springer Science & Business Media, 2011.

[65] Nicholas Roy, Joelle Pineau, and Sebastian Thrun. Spoken dialogue management using probabilistic reasoning. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 93–100. Association for Computational Linguistics, 2000.

[66] Sherry Ruan, Jacob O Wobbrock, Kenny Liou, Andrew Ng, and James Landay. Speech is 3x faster than typing for english and mandarin text entry on mobile devices. *arXiv preprint arXiv:1608.07323*, 2016.

[67] Guergana K Savova, James J Masanz, Philip V Ogren, Jiaping Zheng, Sunghwan Sohn, Karin C Kipper-Schuler, and Christopher G Chute. Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association*, 17(5):507–513, 2010.

[68] Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 149–152. Association for Computational Linguistics, 2007.

[69] Pararth Shah, Dilek Hakkani-Tür, Gokhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry Heck. Building a conversational agent overnight with dialogue self-play. *arXiv preprint arXiv:1801.04871*, 2018.

[70] Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research*, 16:105–133, 2002.

[71] Robert Speer and Catherine Havasi. Representing general relational knowledge in conceptnet 5. In *LREC*, pages 3679–3686, 2012.

[72] Josef Steinberger and Karel Jezek. Using latent semantic analysis in text summarization and summary evaluation. In *Proc. ISIM'04*, pages 93–100, 2004.

[73] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[74] Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307, 2011.

[75] Blaise Thomson and Steve Young. Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems. *Computer Speech & Language*, 24(4):562–588, 2010.

[76] Ivan Titov and Ryan McDonald. Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th international conference on World Wide Web*, pages 111–120. ACM, 2008.

[77] Ivan Titov and Ryan T McDonald. A joint model of text and aspect ratings for sentiment summarization. In *ACL*, volume 8, pages 308–316, 2008.

[78] Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

[79] Panayiotis Tsaparas, Alexandros Ntoulas, and Evimaria Terzi. Selecting a comprehensive set of reviews. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–176. ACM, 2011.

[80] Peter D Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics, 2002.

[81] Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*, 2016.

[82] Jianshu Weng, Chunyan Miao, and Angela Goh. Protecting online rating systems from unfair ratings. In *International Conference on Trust, Privacy and Security in Digital Business*, pages 50–59. Springer, 2005.

[83] Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413, 2013.

[84] Jason D Williams and Steve Young. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422, 2007.

[85] Laurence A Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.

[86] Neal E Young. K-medians, facility location, and the chernoff-wald bound. *arXiv preprint cs/0205047*, 2002.

[87] Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174, 2010.

[88] Li Zhou, Jianfeng Gao, Di Li, and Heung-Yeung Shum. The design and implementation of xiaoice, an empathetic social chatbot. *arXiv preprint arXiv:1812.08989*, 2018.